Oracle® Revenue Management and Billing

Version 2.9.0.0.0

File Upload Interface Implementer’s Guide

Revision 4.1

F34734-01

August, 2020

Oracle Revenue Management and Billing File Upload Interface Implementer's Guide

F34734-01

Preface

About This Document

This document provides a detailed explanation of ORMB approach for Data Conversion and Integration. It describes parameters related to File Upload Interface Master Configuration and explains how to perform important tasks using File Upload Interface. This Guide supplements the information provided in *File Upload Interface User Guide* and *File Upload Interface Batch Execution Guide*.

Intended Audience

This document is intended for the following audience:

- End-Users
- Development Team
- Consulting Team
- Implementation Team

Organization of the Document

The information in this document is organized into the following sections:

Section No.	Section Name	Description
Section 1	ORMB Approach for Data Conversion on Premise	Explains the ORMB Approach for Data Conversion on premise. It also provides the steps to be followed for converting data using ORMB.
Section 2	File Upload Interface	Provides an overview of File Upload Interface
Section 3	ORMB Approach for Data Conversion using FUI	Explains the ORMB Approach for Data Conversion using File Upload Interface. It also provides the steps to be followed for converting data using File Upload Interface.
Section 4	File Upload Interface Master Configuration	Describes important parameters related to File Upload Interface Master Configuration
Section 5	Creating File Request Type	Lists the steps to create file request type
Section 6	Working with File Request Type	Explains different fields in File Request Type zone and tasks which you can perform using File Upload Interface
Section 7	Updating records marked with 'Error' or 'Pending' status	Lists steps to update record status using File Upload Dashboard
Section 8	Transforming Data	Identifies important concepts related to transforming data

Section No.	Section Name	Description
Section 9	File Management System	List of files that are ready to be uploaded in ORMB system
Section 10	Rule Configuration for SFTP Poller File Upload Batch	Rule engine details used in SFTP Poller batch for file polling and uploading in ORMB system

Related Documents

You can refer to the following documents for more information:

Document	Description
Oracle Revenue Management and Billing Banking User Guide	Lists and describes various banking features in Oracle Revenue Management and Billing. It also describes all screens related to these features and explains how to perform various tasks in the application.
File Upload Interface User Guide	Helps you configure File Upload Interface.
File Upload Interface Batch Execution Guide	Explains the batches to be executed while performing various tasks in File Upload Interface.

Conventions

The following conventions are used across the document:

Convention	Meaning
boldface	Boldface indicates graphical user interface elements associated with an action, or terms defined in the text.
italic	Italic indicates a document or book title.
`monospace`	Monospace indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or information that an end-user needs to enter in the application.

Change Log

Revision	Last Update	Updated Section	Comments
4.1	19-Jan-2022	-	Added Bookmarks in PDF

Contents

1. ORMB Approach for Data Conversion on Premise 1
1.1 Prerequisites 1
1.2 Steps for ORMB Conversion 1
1.3 ORMB Conversion Flow Diagram 2
1.4 Disadvantages 2
2. Introduction to File Upload Interface 3
2.1 Pre-staging schema not required 3
3. ORMB Approach for Data Conversion and Integration using File Upload Interface 4
3.1 Prerequisites 4
3.2 Approaches for ORMB Conversion and Integration Process 4
3.2.1 ORMB Conversion and Integration without Data Transformation 4
3.2.2 ORMB Conversion and Integration with Data Transformation 5
4. FUI Master Configuration 6
4.1 Parameters related to FUI Master Configuration 6
4.1.1 Validate Checksum 7
4.1.2 Validate Duplicate File Name 7
4.1.3 Audit Log 7
4.1.4 Archive File 8
4.1.5 Archive File Location 8
4.1.6 Archive Error File Location 8
4.1.7 File Encryption 8
4.1.8 File Decryption Algorithm 9
4.1.9 Cipher Type 9
4.1.10 Decryption Key 14
4.1.11 Upload File Directory 14
5. Creating File Request Type 15
5.1 Creating File Request Type without Data Transformation 15
5.2 Creating File Request Type with Data Transformation 16
6. Working with File Request Type 18
6.1 Uploading and Processing Records using Single Batch Execution 18
6.1.1 File Extensions used for Reading Uploaded Files on SFTP server 18
6.1.2 Rollback all Processed Records in case of Single Record Failure 18
6.1.3 Validating Processing Details 19
6.1.4 Header and Footer Details 20
6.1.5 Ignoring or Skipping Duplicate Records in a File 22
6.1.6 Marking the Default Status of Failed Records to "Retry" for Reprocessing 24
6.1.7 Overriding Service Level Operation 25

6.1.8 Group Duplicate Records 25
6.1.9 Skipping Service Execution 26
6.1.10 Executing Dependent Service 26
6.1.11 Deferring Completion of Processed Request 26
6.1.12 Supporting Client Defined Date Format 27
6.1.13 Usage of Other Display Profile 27
6.1.14 Supporting multiple date formats 28
6.1.15 Viewing TO DO Entries for Failed Files 29
6.1.16 Post Processing Algorithm Support for File Request 30
6.1.17 Auto Generate Field Transformation 31
6.2 Approval Workflow configuration for File Upload Interface 31
6.2.1 Create Approval Workflow Group 31
6.2.2 Create Approval Workflow Chain 32
6.2.3 Create Approval Workflow Criterion Type 33
6.2.4 Create Approval Workflow Group Chain Linkage 34
6.2.5 Create Approval Workflow Settings 34
6.2.6 Update "File Upload Approval Required" flag as true 35
7. Updating Records marked with 'Error' or 'Pending' status 36
7.1 File Upload Dashboard 36
7.2 File Request Status Update Batch 37
8. Transforming Data 38
8.1 Transforming File Record in Client Supported Format into ORMB Conform XML 38
8.2 Configuring a Default Value in Data Transformation 38
8.2.1 Applying Default Values Set in File Validation Algorithm to a Field 39
8.3 Mapping Data Line or Record in a File 40
8.3.1 Mapping XML/JSON Data Line or Record in a File 43
8.4 Mapping File Sequence 43
8.5 Using Map Field XPath to Transform Records 44
8.6 Link Parent Service Output with Child Service Input 45
8.7 Validating Input Values 46
9. File Management System 47
9.1 Overview of files uploaded on SFTP server 47
10. Rule configuration for SFTP poller batch 48
10.1 Rule Type for polling files on SFTP server 48
10.2 Rule Type for failure notification 51
10.3 Create Rule using Rule Type 54
10.3.1 Rule Output Parameters 55
10.3.2 Criteria 56
11. Functions used for FRT configuration 57

11.1 FETCH functions ... 57
11.1.1 Fetch Account ... 58
11.1.2 Fetch Account Characteristic ... 60
11.1.3 Fetch Account Party UID ... 61
11.1.4 Fetch Adjustment Contract ... 62
11.1.5 Fetch Characteristic Entity Flag ... 64
11.1.6 Fetch Characteristic Type Flag ... 64
11.1.7 Fetch Contract ... 65
11.1.8 Fetch Extendable Lookup value ... 66
11.1.9 Fetch Invoice Construct ... 67
11.1.10 Fetch Lookup ... 68
11.1.11 Fetch Parent Person ... 69
11.1.12 Fetch Price Assignment ... 70
11.1.13 Fetch Person ... 71
11.1.14 Fetch Person Characteristic ... 73
11.1.15 Fetch Price Item Contract ... 75
11.1.16 Fetch Price List ... 76
11.1.17 Fetch Price List Characteristic ... 77
11.1.18 Fetch Price List Assignment ID ... 78
11.1.19 Fetch Person Party UID ... 80
11.1.20 Fetch Person Relationship ... 81
11.1.21 Fetch Rate Map Id ... 82
11.1.22 Fetch User ... 83
11.1.23 Check Access Group ... 84
11.2 COPY function ... 85
11.2.1 INPUT Parameter details: ... 85
11.3 String Handling ... 89
11.3.1 Substring ... 90
11.3.2 Prefix the input string ... 93
11.3.3 Substring from search string ... 93
11.3.4 Convert string to uppercase ... 94
11.3.5 Convert string to lowercase ... 94
11.3.6 Right trim the input string ... 94
11.3.7 Left trim the input string ... 94
11.3.8 Trim the input string ... 95
11.3.9 CONCAT ... 95
11.3.10 ISNUMBER ... 96
11.3.11 LENGTH ... 96
11.3.12 ISUTF8 ... 97

11.3.13 CONTAINS 97
11.3.14 STARTSWITH 97
11.3.15 ENDSWITH 98
11.4 SCRIPT function 99
11.5 TO DO Generation 101
11.6 EVAL function 103
11.7 DUPLICATE function 104
11.8 Get Feature Configuration value 104
11.9 HEADER function 105

1. ORMB Approach for Data Conversion on Premise

Enterprise information systems comprise of a variety of data storage systems, which vary in complexity and in the ways they access internal data.

- **Shared Database** - All applications that you are integrating read data directly from the same database.
- **Maintain Data Copies** - Maintain copies of the application's database so that other applications can read the data (and potentially update it).
- **File Transfer** - Make the data available by transporting a file that is an extract from the application's database so that other applications can load the data from the files.
- **Service Integration** - Real time integration using SOAP/REST services.

ORMB uses **File Transfer** approach for data conversion and data integration.

1.1 Prerequisites

To convert data using ORMB on premise, you should have:

1. Conversion tool kit
2. Pre-staging, staging and production schema
3. Stored procedures to map (transform) and transfer data from **pre-staging to staging** and **staging to production**

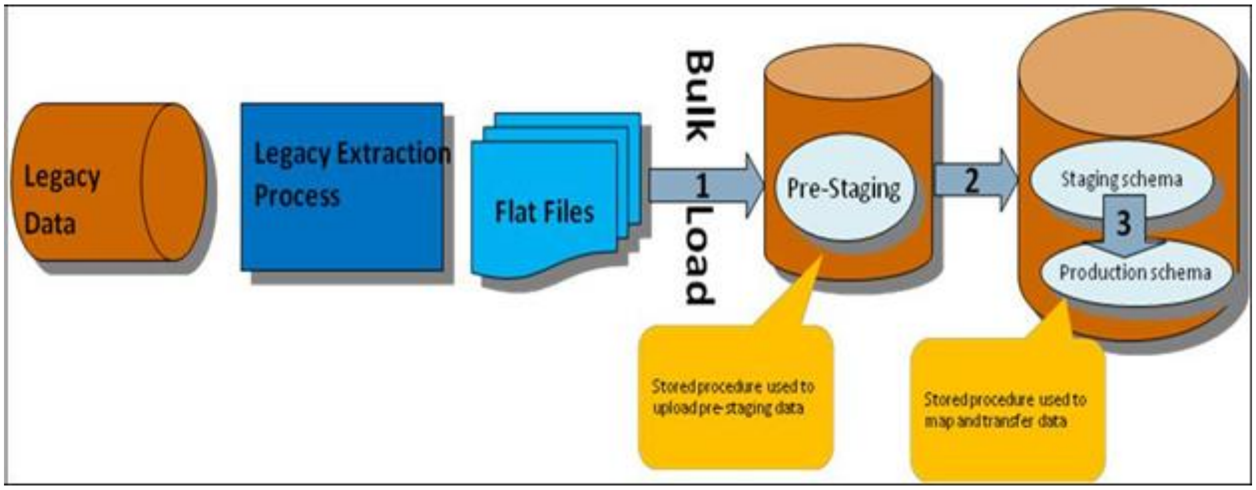
1.2 Steps for ORMB Conversion

To convert data, you need to follow the below steps:

1. The legacy data has to be manually relocated to a database for ConversionMapper to access the data before converting into ORMB table structure.
2. Create a Pre-Staging schema.
3. Define tables to refer to legacy tables external to ORMB. These tables (that contain legacy data) are defined as Input Tables in ConversionMapper.
4. A Bulk Load utility such as SQL*Loader in Oracle is typically used to copy legacy files into ORMB Pre-Staging schema.
5. Create a Staging schema.
6. Instead of using the flat files directly, Oracle tables (Input Tables) that represent the flat files are used to map to ORMB.
7. The legacy tables or "Input Tables" in ConversionMapper MUST exist on a database (in staging schema), so that ConversionMapper can use SQL statements to access the tables.
8. The legacy tables can be defined as tables on the staging schema (STGADM) or as views (also on STGADM) of tables on another schema or database.
9. Stored Procedures are used to map, validate and transfer the data to staging schema.

1.3 ORMB Conversion Flow Diagram

The flow of data between the two systems is illustrated below:


Legacy Data
Legacy Extraction Process
Flat Files
Bulk Load
1
Pre-Staging
2
Staging schema
3
Production schema
Stored procedure used to upload pre-staging data
Stored procedure used to map and transfer data

1.4 Disadvantages

- Data transformation and loading is done using Stored procedures thereby resulting in duplication of data validation and business rules.
- Risk of missing data validation and business rules.

2. Introduction to File Upload Interface

ORMB File Upload Interface provides ability to upload files to staging and map file records through ORMB services. ORMB File Upload Interface invokes ORMB services for each of the record. The ORMB file upload interface provides following benefits:

- Conversion tool kit not required

2.1 Pre-staging schema not required

- ORMB DB access not required for any legacy system
- Data load in ORMB is done using existing ORMB service schemas, thereby no duplication of data validation and business rules
- No risk of missing data validation and business rules
- Supports transformation for files in XML, JSON, Fixed Position, CSV, PSV and TSV formats
- Online system can be used to view the uploaded files and their corresponding processed details

3. ORMB Approach for Data Conversion and Integration using File Upload Interface

This chapter lists the steps to be followed before using File Upload Interface. It also describes approaches you can take when working with ORMB File Upload Interface.

3.1 Prerequisites

To convert data using ORMB using File Upload Interface, you should:

- **Define File Upload Interface master configuration**: For more information on how to define File Upload Interface master configuration, refer File Upload Interface User Guide.
- **Configure File Request Type** with required mapping of existing ORMB Services Business Objects or Business Services or Service Script specific to Transaction Stage Upload service. You need to add date format in `FILE_UPLOAD_DATE_FORMATS` lookup to support client-specific date inputs in Transaction Header.

3.2 Approaches for ORMB Conversion and Integration Process

To convert or integrate files in ORMB file upload interface, you can follow any of the below two approaches:

- ORMB conversion and integration process **without** Data Transformation
- ORMB conversion and integration process **with** Data Transformation

3.2.1 ORMB Conversion and Integration without Data Transformation

To convert or integrate files without Data Transformation, you need to follow below steps:

1. Ensure that the files are in XML format and comply with ORMB service schema.
2. Publish service XMLs conforming to ORMB service schemas that are to be used for conversion.
3. If validations for Header, Footer or Checksum are required, then you need to write an algorithm deriving `FileValidationAlgorithmSpot`.
4. If any preprocessing is required before invoking Business Object or Business Service or Service Script service, then you need to implement an algorithm deriving `FileRequestPreProcessingAlgorithmSpot`.
5. Execute File Transform and Upload batch (**C1-FTRAN**). This reads flat file records and upload in file upload staging. For more information on C1-FTRAN batch, refer to File Upload Interface Batch Execution Guide.
6. Execute File Request Processing batch (**C1-FREQP**). This reads the records from staging and process those using Services configured in file request type. For more information on C1- FREQP batch, refer File Upload Interface Batch Execution Guide.

7. File upload and processing steps can also be done by executing only C1-FTRAN batch. You are required to set "File Upload and Process" flag as true in File Request Type configuration.
8. Perform required fixes in File Request Type Configuration or File Details.

Note: During conversion process, conversion cycles can be run on UAT environment while the system is being tested on the production schema.

3.2.2 ORMB Conversion and Integration with Data Transformation

To convert or integrate files with Data Transformation:

File upload supports files in XML, JSON, CSV, PSV, TSV or Fixed Position formats.

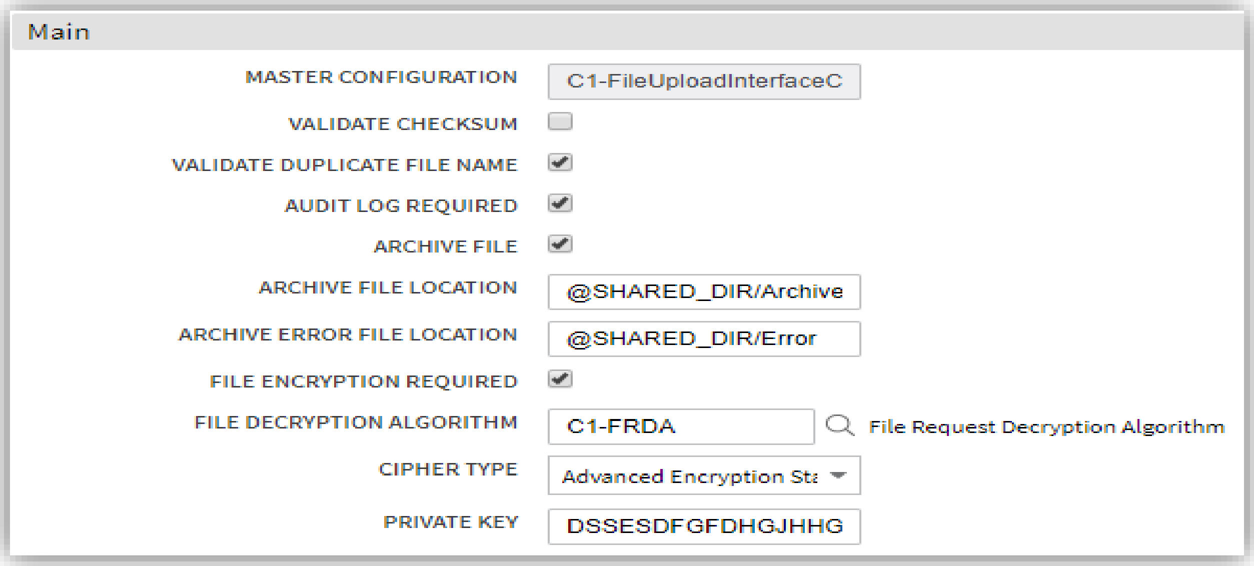
1. Data Transformation algorithm can be implemented by deriving "FileRequestTranformationAlgorithmSpot".
2. Data Transformation will be done using Data Transformation configuration in its file request type.
3. If validations for Header, Footer or Checksum are required, then implement an algorithm deriving "FileValidationAlgorithmSpot".
4. If any preprocessing is required before invoking Business Object or Business Service or Service Script service, then implement an algorithm deriving "FileRequestPreProcessingAlgorithmSpot".
5. Execute File Transform and Upload batch (C1-FTRAN). This will read the flat file records and transform those using "Data Transformation" algorithm and upload in file upload staging.
6. Execute File Request Processing batch (C1-FREQP). This will read the records from staging and process those using Service configured in File Request Type.
7. File transform, upload and processing steps can also be done by executing only C1-FTRAN batch. You are required to set "File Upload and Process" flag as true in File Request Type configuration.
8. Perform required fixes either in File Request Type Configuration or file details.

Note: During conversion process, conversion cycles can be run on UAT environment while the system is being tested on the production schema.

4. FUI Master Configuration

This configuration is referred to in file upload for file decryption, archival of file, and audit logging. To view and edit file upload configuration, you need to do the following:

1. From the Admin menu, select M and then click Master Configuration. The Master Configuration window appears.
2. Click Edit corresponding to the File Upload Interface Configuration. The Master Configuration window appears.



Main

MASTER CONFIGURATION	C1-FileUploadInterfaceC
VALIDATE CHECKSUM	☐
VALIDATE DUPLICATE FILE NAME	✓
AUDIT LOG REQUIRED	✓
ARCHIVE FILE	✓
ARCHIVE FILE LOCATION	@SHARED_DIR/Archive
ARCHIVE ERROR FILE LOCATION	@SHARED_DIR/Error
FILE ENCRYPTION REQUIRED	✓
FILE DECRYPTION ALGORITHM	C1-FRDA File Request Decryption Algorithm
CIPHER TYPE	Advanced Encryption Sta
PRIVATE KEY	DSSESDFGFDHGJHHG

Figure 1: File Upload Interface Configuration - Master Configuration

4.1 Parameters related to FUI Master Configuration

This section lists and describes following important parameters related to File Upload Interface Master Configuration:

- Validate Checksum
- Validate Duplicate File Name
- Audit Log
- Archive File
- Archive File Location
- Archive Error File Location
- File Encryption
- File Decryption Algorithm
- Cipher Type
- Decryption Key

4.1.1 Validate Checksum

This flag decides whether to check the integrity of the file before staging file contents in ORMB system or not. The Validate Checksum parameter is a check box:

- If **selected**, checksum validation is performed for all the uploaded files
- If **not selected**, checksum validation is skipped

A few pointers regarding Checksum validation:

- Every uploaded file must have a corresponding `<FILE_NAME>.checksum` file on SFTP server.
- A File validation algorithm must be mapped with every File Request Type, which is derived from `FileHeaderValidationAlgorithmSpot`.
- The Checksum validation logic is implemented in the File validation algorithm, which is also used for Header and Footer validations.

File Validation algorithm is derived from "FileHeaderValidationAlgorithmSpot".

A sample algorithm `FileHeaderValidationAlgorithm_Impl` is available with:

- Header validation – Number of records are checked
- Checksum validation – Done using **MD5** algorithm type

For checksum validation, if you want to use the default implementation in any specific File validation algorithm, then it can be done by invoking `calculateChecksum(String fileString, String algoName)` function in `FileRequestProcessBusinessComponent_Impl` business component.

Note: For uploaded file, it is not required to have corresponding <FILE_NAME>.checksum file.

4.1.2 Validate Duplicate File Name

This flag is used to decide the validation of duplicate file name before uploading a file. The Validate Duplicate File Name parameter is a Check box with valid values True and False.

- If **True**, if a file with same name exists in staging, then the file is uploaded
- If **False**, file with same name is uploaded

4.1.3 Audit Log

This flag decides whether to log corresponding status changes of an individual file request after processing. The Audit Log parameter is a Check box with valid values True and False.

- If **True**, status transitions for all file requests are maintained in ORMB
- If **False**, status transitions for the file requests are not performed

Logging is done in `CI_FILE_REQUEST_DTL_MSG` table with `LOG_ENTRY_TYPE_FLG` value as **F1ST**. `CI_FILE_REQUEST_DTL_MSG` table is also used for error logging with `LOG_ENTRY_TYPE_FLG` value as **F1EX**.

4.1.4 Archive File

This flag decides whether to relocate the file to another location after processing on SFTP server. Here, location refers to the path mentioned in "Archive File Location" or "Archive Error File Location". The Archive File parameter is a Check box with valid values True and False.

- If **True**, files are moved to another location. There are two scenarios:
  - If file is uploaded successfully, the files are moved to a defined archive file location.
  - If file upload fails, the files are moved to a defined archive error file location.
- If **False**, the files remain at the same location

4.1.5 Archive File Location

It is used to specify the file path used for archiving the file. The successfully processed file is moved to this location. The Archive File Location parameter is a field where you can enter the location.

- File Location should always be a combination of logical path and relative path and prefixed with either `INSTALL_DIR` or `SHARED_DIR`. For example, if you want to define file location as "/scratch/rmbbuild/sftpFile", then it will be **INSTALL_DIR/ sftpFile** where `INSTALL_DIR` value is defined as "/scratch/rmbbuild".
  - Define `INSTALL_DIR` variable value against `spl.runtime.environ.SPLEBASE` property in spl.properties file.
  - `SHARED_DIR` variable is a shared storage path on the application environment and has a static value.
- The defined location is appended by the corresponding File Request Type. For example, Archive Error File Location is **INSTALL_DIR/FilesUploaded**, where INSTALL_DIR path is **/scratch/rmbbuild**. If a batch is executed for `ADD_PERSON` file request type, then files are moved to **/scratch/rmbbuild/FilesUploaded/ADD_PERSON/**

4.1.6 Archive Error File Location

It is used to specify the file path used for archiving the error files. The files with errors will be moved to this location. The Archive Error File Location parameter should be a combination of logical path and relative path and prefixed with either `INSTALL_DIR` or `SHARED_DIR`. The defined location will be always appended by the corresponding File Request Type. For example, Archive Error File Location is **INSTALL_DIR/ErrorFiles** and INSTALL_DIR path is **/scratch/rmbbuild**. If a batch is executed for `ADD_PERSON` file request type, then files are moved to **/scratch/rmbbuild/ErrorFiles/ADD_PERSON/**

4.1.7 File Encryption

This flag decides whether to first decrypt and then extract the files on SFTP server. The File Encryption parameter is a Check box with values: True or False.

- If **True**, files on SFTP server are decrypted using File Decryption Algorithm, then extracted and processed to upload the file data in ORMB staging.
- If **False**, files on SFTP server are uploaded without decryption.

For using the file encryption parameter, a File Decryption Algorithm should be available. The algorithm is derived from `FileRequestDecryptionAlgorithmSpot` and has implementation commands required for getting the decrypted file.

A sample algorithm `FileRequestDecryptionAlgorithm_Impl` is provided with ORMB application and:

- Gets the decryption key for defined `com.oracle.ouaf.system.keystore.file` alias in **ouaf_keystore** file
- Gets the decrypted file using decryption key

4.1.8 File Decryption Algorithm

It defines the algorithm to be used for decryption of a file. You can search for the File Decryption Algorithm `FileRequestDecryptionAlgorithmSpot`. This algorithm must have an implementation for decrypting the input File.

- It gets the secret key stored against

  `com.oracle.ouaf.system.keystore.passwordFileName` alias and uses secret key to access ouaf_keystore file.
- It gets the decryption key stored against `com.oracle.ouaf.system.keystore.file` alias in ouaf_keystore file.
- The decryption key decrypts and returns the file string.

4.1.9 Cipher Type

This is an algorithm type used to get the decryption key. The Cipher Type parameter has a drop down list with values:

- Advanced Encryption Standards
- Data Encryption Standard
- Pretty Good Privacy(PGP)

Advanced Encryption Standards(AES)

AES is a symmetric key encryption, which involves the use of only one secret key to cipher and decipher information. Secret key is captured in ORMB using File Upload Interface master configuration UI.

Data Encryption Standards(DES)

The Data Encryption Standard (DES) is a symmetric-key method of data encryption. DES works by using the same key to encrypt and decrypt a message, so both the sender and the receiver must know and use the same private key.

Pretty Good Privacy(PGP)

PGP uses hybrid cryptosystem by combining symmetric-key encryption and public-key (asymmetric-key) encryption.

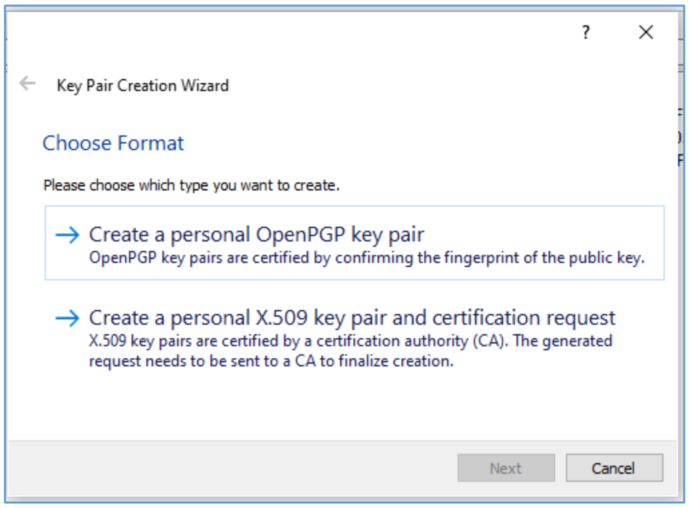
Follow the steps below for encryption and decryption of a file using PGP:

1. Client generates a private-public key pair using standard PGP key generation utility, i.e. "Kleopatra".

   a. Steps to generate key pair:

      i. Open the utility Kleopatra and navigate to File, New Key Pair. This opens the Key Pair Creation Wizard.

      ii. In the wizard, select "Create a Personal OpenPGP key pair" option and click Next.



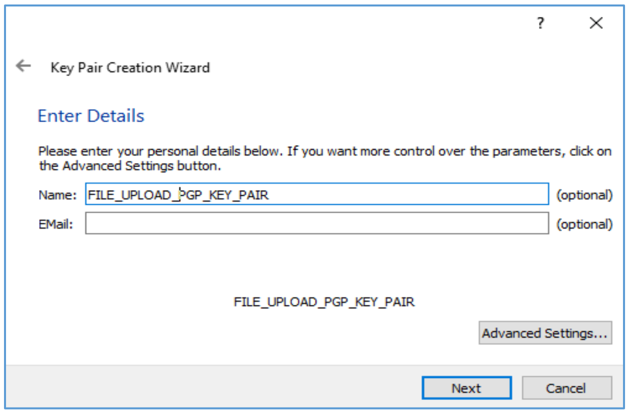
Choose Format

Please choose which type you want to create.

→ Create a personal OpenPGP key pair
OpenPGP key pairs are certified by confirming the fingerprint of the public key.

→ Create a personal X.509 key pair and certification request
X.509 key pairs are certified by a certification authority (CA). The generated request needs to be sent to a CA to finalize creation.

      iii. Enter name and click Next.



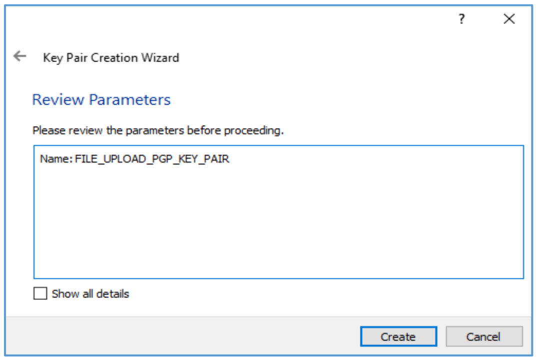
Enter Details

Please enter your personal details below. If you want more control over the parameters, click on the Advanced Settings button.

Name: FILE_UPLOAD_PGP_KEY_PAIR (optional)
EMail: (optional)

FILE_UPLOAD_PGP_KEY_PAIR

      iv. Review the details and click Create.



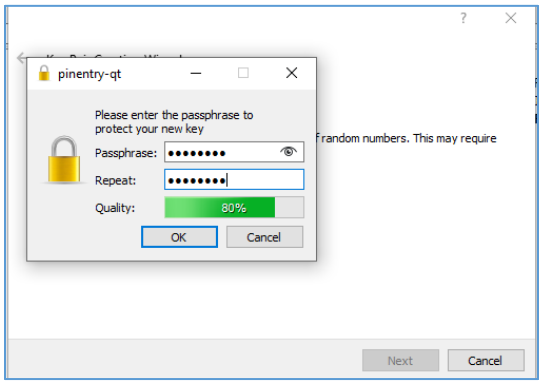
Review Parameters

Please review the parameters before proceeding.

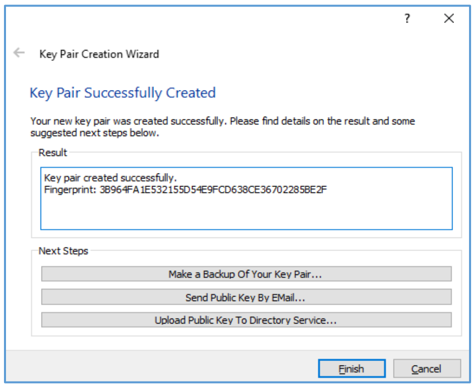
Name: FILE_UPLOAD_PGP_KEY_PAIR

v. When prompted, enter a passphrase (minimum 8 characters long) and click OK.

Note: Make a note of this passphrase as you need to provide it during File Upload Interface master configuration for PGP support.


pinentry-qt
Please enter the passphrase to protect your new key
Passphrase: ••••••••
Repeat: ••••••••
Quality: 80%
OK
Cancel
random numbers. This may require
Next
Cancel

vi. Click Finish. The key pair is successfully generated.



Key Pair Successfully Created

Your new key pair was created successfully. Please find details on the result and some suggested next steps below.

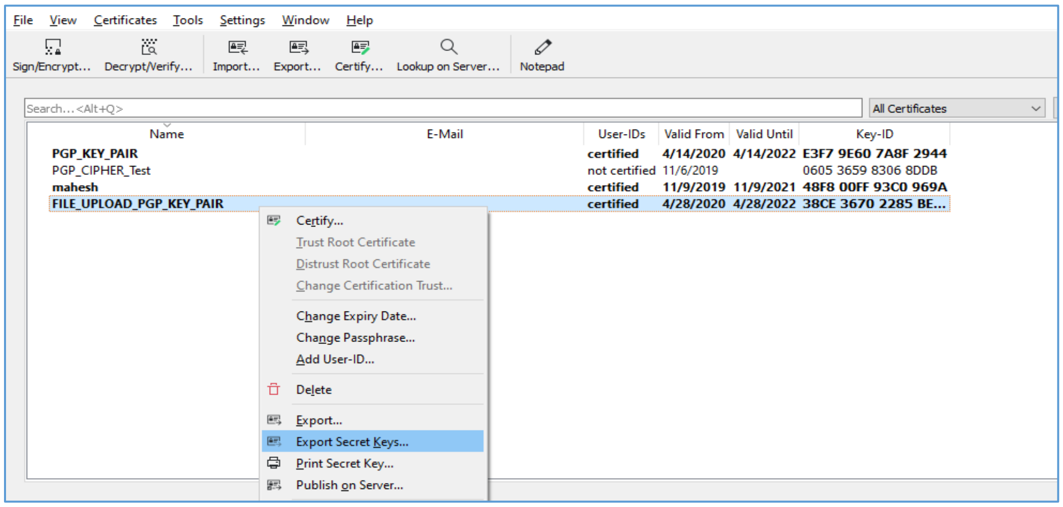
Result

Key pair created successfully.
Fingerprint: 3B964FA1E532155D54E9FCD638CE36702285BE2F

Next Steps
Make a Backup Of Your Key Pair…
Send Public Key By EMail…
Upload Public Key To Directory Service…
Finish
Cancel

b. Follow the steps below to export Private/Secret key to a file:

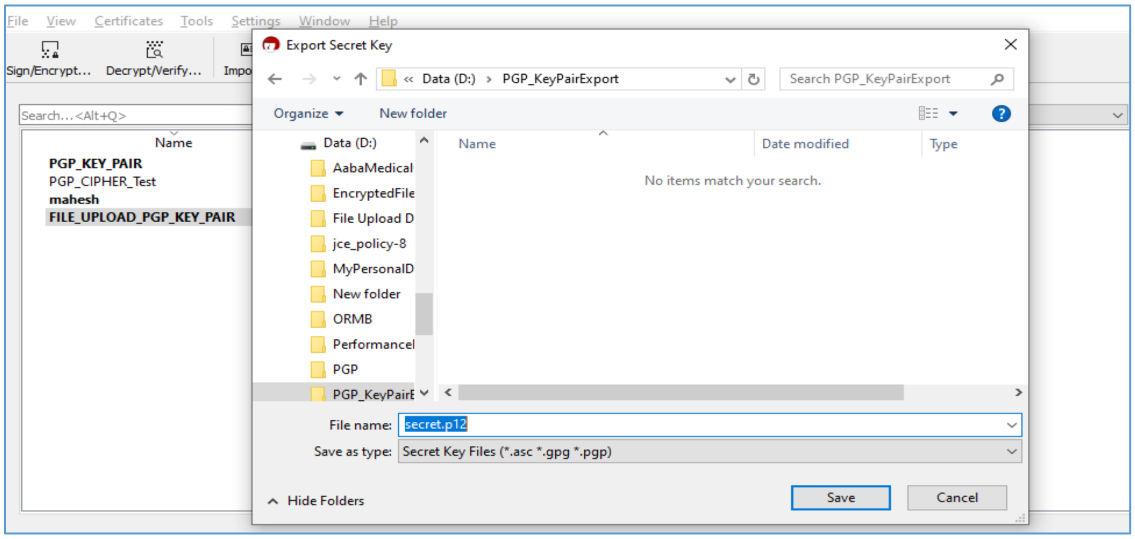
i. Click on Key Pair Name for which you want to export the secret key; i.e. 'FILE_UPLOAD_KEY_PAIR' and select Export Secret Keys option.


File View Certificates Tools Settings Window Help
Sign/Encrypt… Decrypt/Verify… Import… Export… Certify… Lookup on Server… Notepad
Search…<Alt+Q>
All Certificates

Name	E-Mail	User-IDs	Valid From	Valid Until	Key-ID
PGP_KEY_PAIR		certified	4/14/2020	4/14/2022	E3F7 9E60 7A8F 2944
PGP_CIPHER_Test		not certified	11/6/2019		0605 3659 8306 8DDB
mahesh		certified	11/9/2019	11/9/2021	48F8 00FF 93C0 969A
FILE_UPLOAD_PGP_KEY_PAIR		certified	4/28/2020	4/28/2022	38CE 3670 2285 BE…

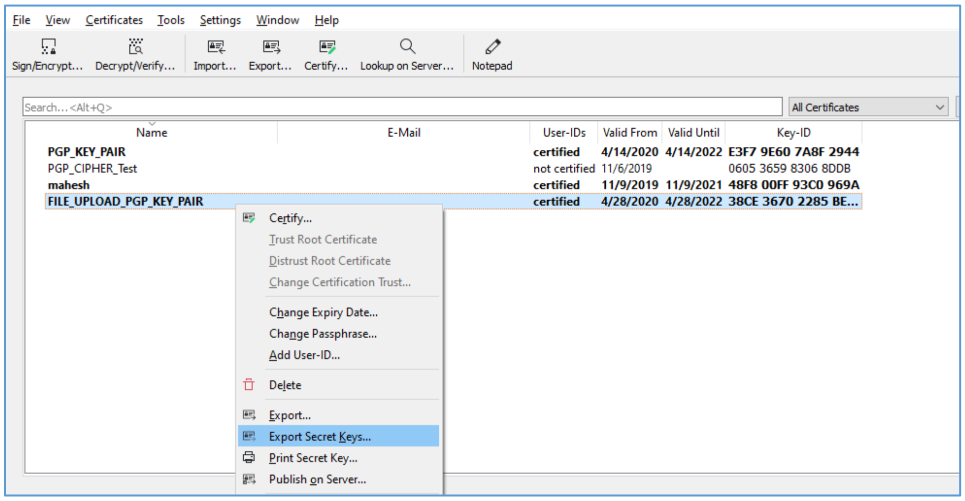
Certify…
Trust Root Certificate
Distrust Root Certificate
Change Certification Trust…
Change Expiry Date…
Change Passphrase…
Add User-ID…
Delete
Export…
Export Secret Keys…
Print Secret Key…
Publish on Server…

ii. Store the file on your local drive with **.p12** extension.



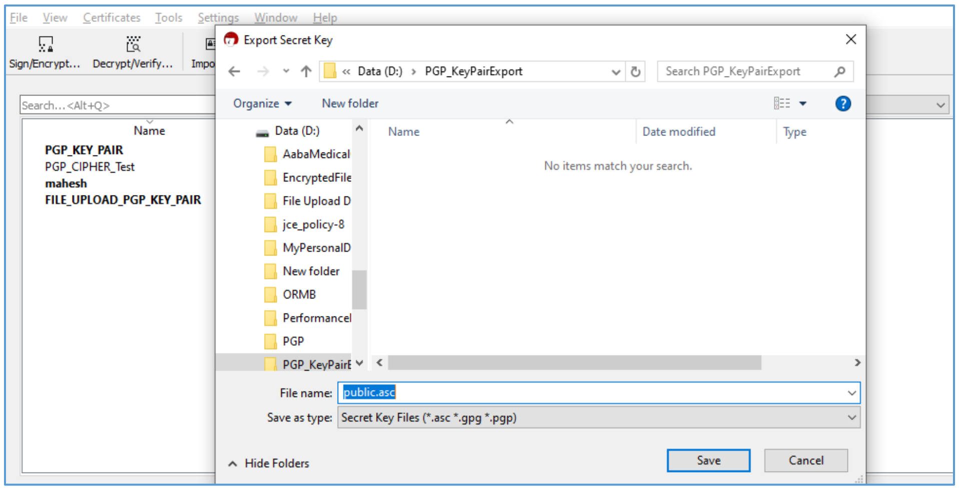
c. Follow the steps below to export Public Key in a file:

   i. Click on Key Pair Name for which you want to export the secret key; i.e. FILE_UPLOAD_KEY_PAIR and select Export Secret Keys option.


File View Certificates Tools Settings Window Help
Sign/Encrypt... Decrypt/Verify... Import... Export... Certify... Lookup on Server... Notepad
Search... <Alt+Q>
All Certificates
Certify...
Trust Root Certificate
Distrust Root Certificate
Change Certification Trust...
Change Expiry Date...
Change Passphrase...
Add User-ID...
Delete
Export...
Export Secret Keys...
Print Secret Key...
Publish on Server...

Name	E-Mail	User-IDs	Valid From	Valid Until	Key-ID
PGP_KEY_PAIR		certified	4/14/2020	4/14/2022	E3F7 9E60 7A8F 2944
PGP_CIPHER_Test		not certified	11/6/2019		0605 3659 8306 8DDB
mahesh		certified	11/9/2019	11/9/2021	48F8 00FF 93C0 969A
FILE_UPLOAD_PGP_KEY_PAIR		certified	4/28/2020	4/28/2022	38CE 3670 2285 BE...

   ii. Store the file on your local drive with **.asc** extension.

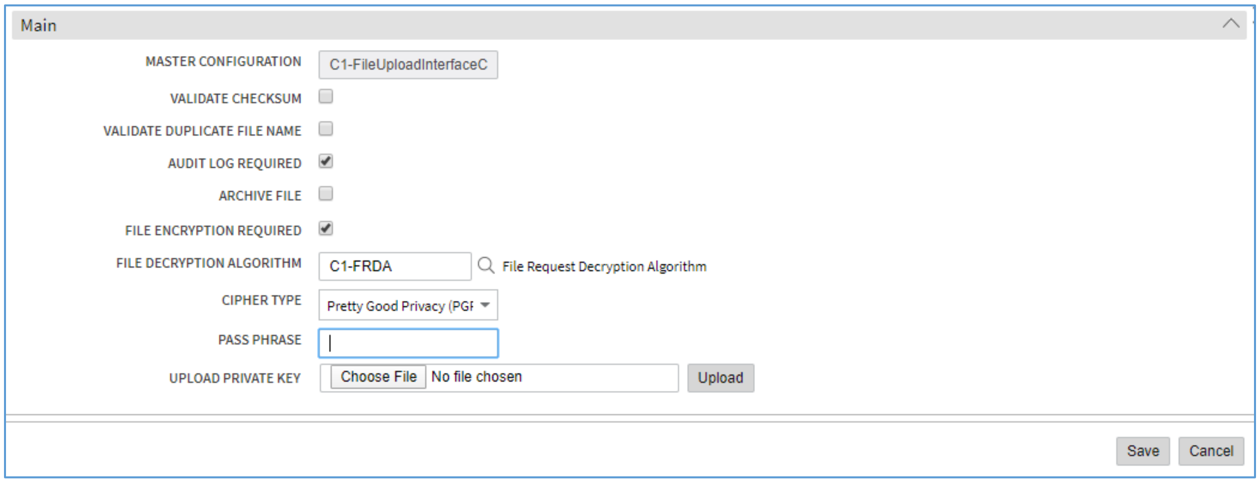


File name: public.asc

Save as type: Secret Key Files (*.asc *.gpg *.pgp)

iii. Legacy system uses this public key for file encryption.

2. Client has to upload the paired private key file using File Upload Interface master configuration UI.
   a. Upload this secret key from File Upload Interface master configuration
      i. Go To Admin Menu>>Master Configuration>>File Upload Interface Configuration
      ii. Click on Edit Button
      iii. Select File Encryption Required checkbox
      iv. Select File Decryption Algorithm: C1-FRDA
      v. Select Cipher Type – Pretty Good Privacy (PGP)
      vi. Enter Passphrase used while creating the key pair
      vii. Click on upload button to upload the secret.p12 file



MASTER CONFIGURATION	C1-FileUploadInterfaceC
VALIDATE CHECKSUM	☐
VALIDATE DUPLICATE FILE NAME	☐
AUDIT LOG REQUIRED	✓
ARCHIVE FILE	☐
FILE ENCRYPTION REQUIRED	✓
FILE DECRYPTION ALGORITHM	C1-FRDA File Request Decryption Algorithm
CIPHER TYPE	Pretty Good Privacy (PGI
PASS PHRASE	
UPLOAD PRIVATE KEY	No file chosen

- Further, using this private key, file upload interface will decrypt and process all the encrypted file's uploaded on SFTP server by legacy system.

Pre-requisite:

- Software - gpg4win-3.1.9
- File upload master configuration required to be done for PGP support.
- **spl-reporting** ear deployed and running on server.
- Mention EJB properties in /splapp/standalone/config/spl.properties file
  ## EJB properties
  ```
  spl.serviceBean.jndi.name = ouaf/servicebean
  spl.ejbContainer.contextFactory =
  weblogic.jndi.WLInitialContextFactory
  spl.ejbContainer.url = t3://whfXXXX.in.oracle.com:7001
  spl.ejbContainer.user = Weblogic_Console_Username
  spl.ejbContainer.password = Weblogic_Console_Password (Encrypted)
  ```
- Copy these EJB properties from /splapp/mpl/config/spl.properties file.
- Update "**spl.ejbContainer.url**" property value from **t3s** to **t3** and update the given port with **non-SSL** port.
- Check the value of property `ouaf.system.fileupload.pgp.pvt.filepath` to see if it has the same private file name that you have uploaded "secret.p12". If required, update the file name.
- Save changes in spl.properties file
- Restart the Threadpool(s).

4.1.10 Decryption Key

This key is used to decrypt the uploaded encrypted files on SFTP server. The Decryption Key parameter holds a decryption key, which is further used to get the decrypted file data. This field value is not stored in the database, but in ORMB keystore with `com.oracle.ouaf.system.keystore.file` alias in ouaf_keystore file.

4.1.11 Upload File Directory

This Upload File Directory holds the root directory path of SFTP server where third party files can be uploaded. All the files in this directory and subdirectories are picked by SFTP File Upload Poller batch and uploaded in ORMB system using Rule Engine and DBMS scheduler batch job. This directory path is also used by File Management System UI to display all available files in the location.

5. Creating File Request Type

File request type is a configuration that allows you to upload files in any format and transform the files in ORMB compliant format.

When creating a file request type, you have two options:

- Create File Request Type without Data Transformation
- Create File Request Type with Data Transformation

5.1 Creating File Request Type without Data Transformation

To create file request type without Data Transformation:

1. Navigate to Admin > F > File Request Type > Add. The File Request Type window appears. This window has following sections:
   - Main
   - Services
   - Messages
   - Data Transformation
2. Enter name of file request type in File Request Type field.
3. Verify if the value for File Format field is Extensible Markup Language.

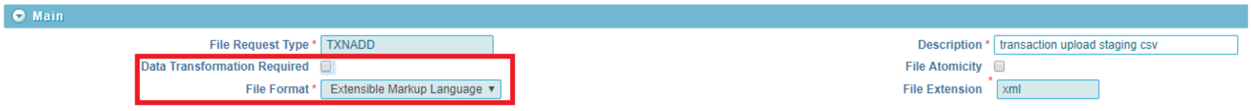
Note: Default value for File Format field is Extensible Markup Language.

- If the value is **Extensible Markup Language**, go to Step 5.
- If File Format has any other value,

   b. Select Data Transformation Required check box. The File Format field gets enabled.

   c. Select Extensible Markup Language from the File Format drop-down list.

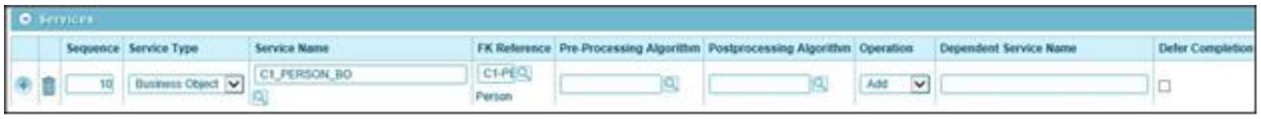
   d. Deselect Data Transformation Required flag and go to Step 5.


Main

File Request Type *	TXNADD	Description *	transaction upload staging csv
Data Transformation Required	☐	File Atomicity	☐
File Format *	Extensible Markup Language	File Extension *	xml

Figure 2: Data Transformation

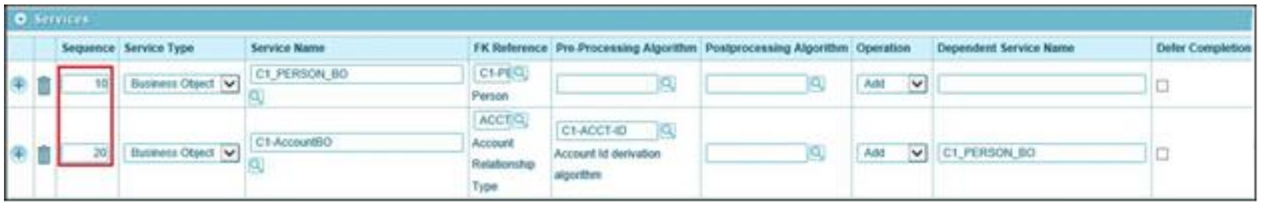
4. Configure at least one service within Service section. This service will be used to process flat file records.


Services

Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Postprocessing Algorithm	Operation	Dependent Service Name	Defer Completion
10	Business Object	C1_PERSON_BO	C1-PE Person			Add		☐

Figure 3: Configuring Service

5. Multiple relational or non-relational services can be configured under single File Request Type. For multiple services, service execution sequence of each record is decided on the basis of given Sequence number.



Services

Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Postprocessing Algorithm	Operation	Dependent Service Name	Defer Completion
10	Business Object	C1_PERSON_BO	C1-PE Person			Add		☐
20	Business Object	C1-AccountBO	ACCT Account Relationship Type	C1-ACCT-ID Account Id derivation algorithm		Add	C1_PERSON_BO	☐

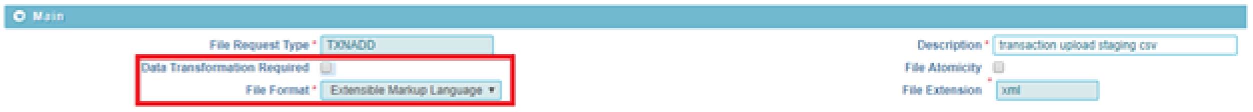
Figure 4: Configuring Multiple Services without Data Transformation

6. Click Save.

5.2 Creating File Request Type with Data Transformation

To create file request type with Data Transformation:

1. Navigate to Admin > F > File Request Type > Add. The File Request Type window appears.
2. Enter name of file request type in File Request Type field.
3. Select Data Transformation Required check box. The File Format field gets enabled.

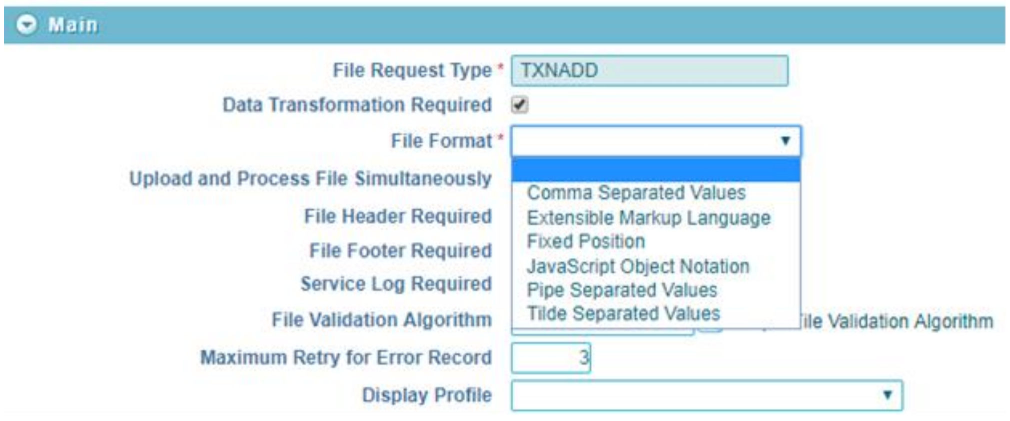


Main

File Request Type * TXNADD
Data Transformation Required ☐
File Format * Extensible Markup Language
Description * transaction upload staging csv
File Atomicity ☐
File Extension * xml

Figure 5: Data Transformation

4. Select the required file format from File Format drop-down list. The valid values are:
   - Comma Separated Values (CSV)
   - Extensible Markup Language (XML)
   - Fixed Position
   - JavaScript Object Notation (JSON)
   - Pipe Separated Values (PSV)
   - Tilde Separated Values (TSV)

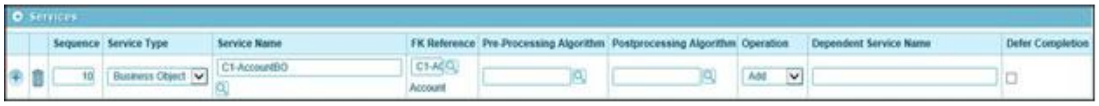


Main

File Request Type * TXNADD
Data Transformation Required ✓
File Format *
Upload and Process File Simultaneously
File Header Required
File Footer Required
Service Log Required
File Validation Algorithm
Maximum Retry for Error Record 3
Display Profile

Figure 6: File Format

5. Configure at least one service within Service section. This service which will be used to process flat file records.

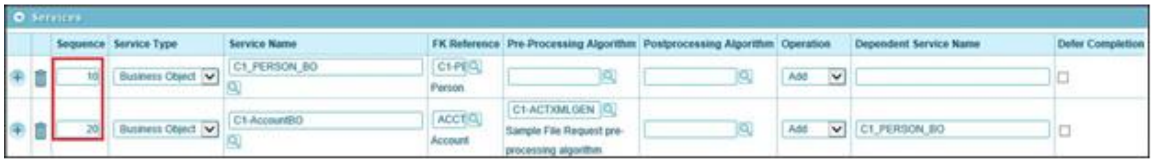


Services

Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Postprocessing Algorithm	Operation	Dependent Service Name	Defer Completion
10	Business Object	C1-AccountBO	C1-AC Account			Add		☐

Figure 7: Service Sequence

6. Multiple relational or non-relational services can be configured under single File Request Type. For multiple services, service execution sequence of each record is decided on the basis of given Sequence number.

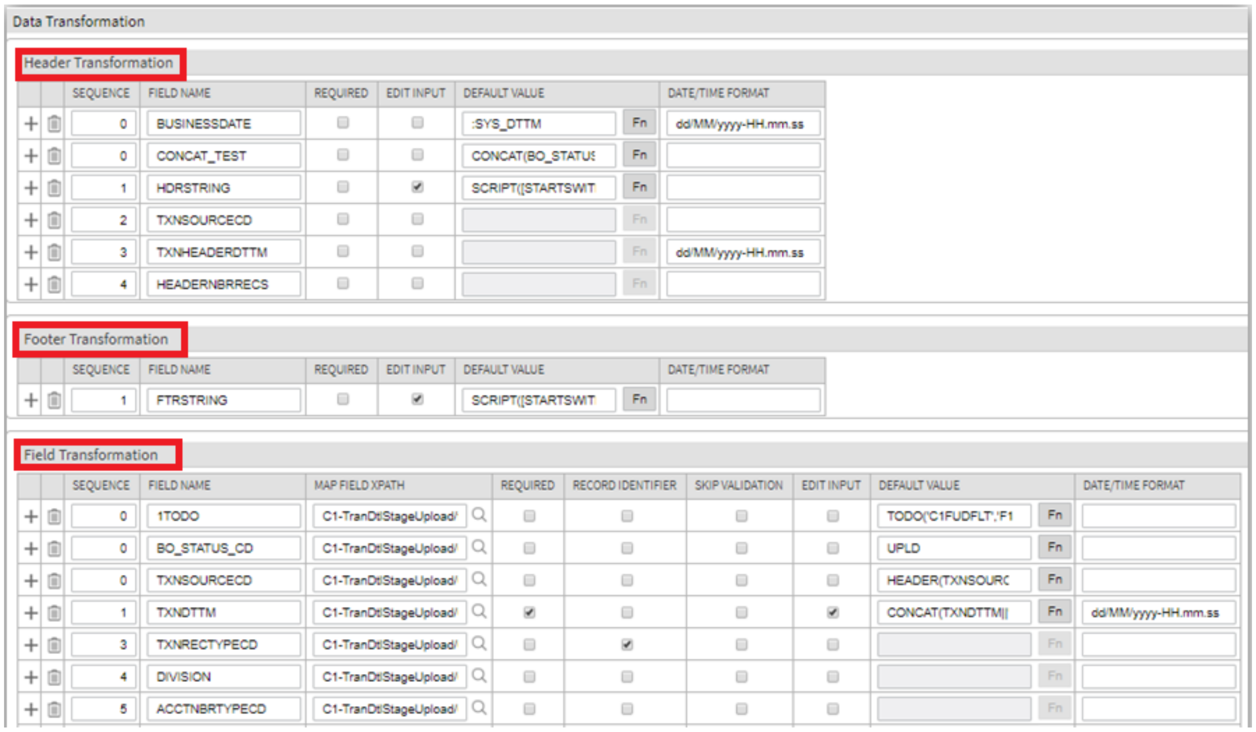


Services

Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Postprocessing Algorithm	Operation	Dependent Service Name	Defer Completion
10	Business Object	C1_PERSON_BO	C1-PE Person			Add		☐
20	Business Object	C1-AccountBO	ACCT Account	C1-ACTXMLGEN Sample File Request pre-processing algorithm		Add	C1_PERSON_BO	☐

Figure 8: Configuring Multiple Services with Data Transformation

7. If data transformation required, then, Data Transformation details need to be configured with respect to its corresponding flat file data line. For more information on file data line, refer section Mapping Data Line or Record in a File.



Data Transformation

Header Transformation

SEQUENCE	FIELD NAME	REQUIRED	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
0	BUSINESSDATE	☐	☐	:SYS_DTTM	dd/MM/yyyy-HH.mm.ss
0	CONCAT_TEST	☐	☐	CONCAT(BO_STATUS	
1	HDRSTRING	☐	☑	SCRIPT([STARTSWIT	
2	TXNSOURCECD	☐	☐		
3	TXNHEADERDTTM	☐	☐		dd/MM/yyyy-HH.mm.ss
4	HEADERNBRRECS	☐	☐		

Footer Transformation

SEQUENCE	FIELD NAME	REQUIRED	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
1	FTRSTRING	☐	☑	SCRIPT([STARTSWIT	

Field Transformation

SEQUENCE	FIELD NAME	MAP FIELD XPATH	REQUIRED	RECORD IDENTIFIER	SKIP VALIDATION	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
0	1TODO	C1-TranDtlStageUpload/	☐	☐	☐	☐	TODO('C1FUDFLT','F1	
0	BO_STATUS_CD	C1-TranDtlStageUpload/	☐	☐	☐	☐	UPLD	
0	TXNSOURCECD	C1-TranDtlStageUpload/	☐	☐	☐	☐	HEADER(TXNSOURC	
1	TXNDTTM	C1-TranDtlStageUpload/	☑	☐	☐	☑	CONCAT(TXNDTTM||	dd/MM/yyyy-HH.mm.ss
3	TXNRECTYPECD	C1-TranDtlStageUpload/	☐	☑	☐	☐		
4	DIVISION	C1-TranDtlStageUpload/	☐	☐	☐	☐		
5	ACCTNBRTYPECD	C1-TranDtlStageUpload/	☐	☐	☐	☐		

Figure 9: Configuring Data Transformation

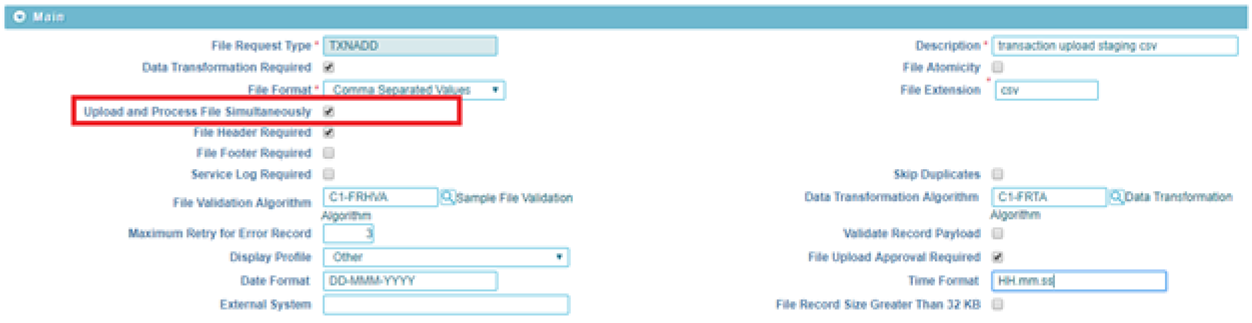
8. Click Save.

6. Working with File Request Type

6.1 Uploading and Processing Records using Single Batch Execution

To upload and process records using single batch execution:

1. Define a new file request type or search for an existing file request type. Select **Upload and Process File Simultaneously** check box. This sets the flag as **True**.



Main

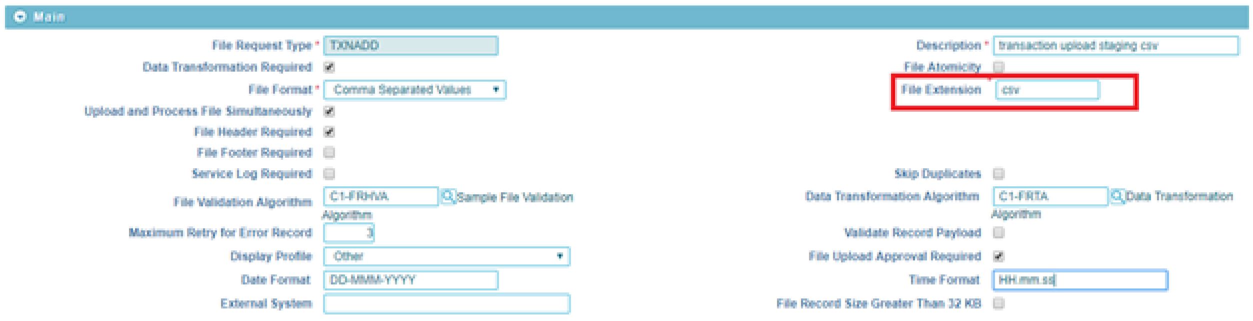
File Request Type *	TXNADD	Description *	transaction upload staging csv
Data Transformation Required	✓	File Atomicity	☐
File Format *	Comma Separated Values	File Extension *	csv
Upload and Process File Simultaneously	✓		
File Header Required	✓		
File Footer Required	☐		
Service Log Required	☐	Skip Duplicates	☐
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	3	Validate Record Payload	☐
Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	☐

Figure 10: Upload and Process Records

2. Execute **C1-FTRAN** batch. This batch uploads the file and start processing of all records in the file.

6.1.1 File Extensions used for Reading Uploaded Files on SFTP server

File Upload interface uses flat files to upload data into ORMB system. These flat files can have extensions like **.txt**, **.csv**, **.dat**, .**xml** etc. The legacy system locates these files on SFTP server and uploads data from it. It is possible that the legacy system locates different files with different extensions at the same location. Hence, the **C1-FTRAN** batch reads only those files that matches extensions configured in File Request Type.



Main

File Request Type *	TXNADD	Description *	transaction upload staging csv
Data Transformation Required	✓	File Atomicity	☐
File Format *	Comma Separated Values	File Extension	csv
Upload and Process File Simultaneously	✓		
File Header Required	✓		
File Footer Required	☐		
Service Log Required	☐	Skip Duplicates	☐
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	3	Validate Record Payload	☐
Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	☐

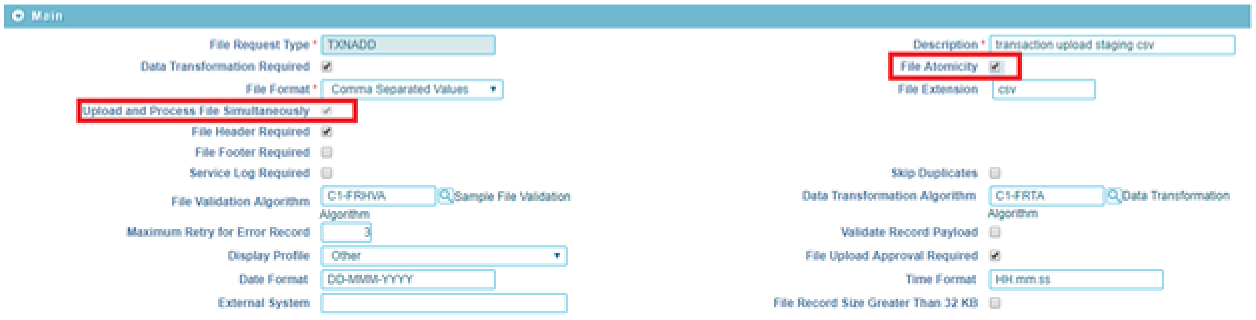
Figure 11: File Extension

6.1.2 Rollback all Processed Records in case of Single Record Failure

In case of single record failure in a file, you can rollback all those processed records by executing **C1-FTRAN** batch.

To rollback processed records, select **File Atomicity** check box against the File Request Type. This automatically selects the **Upload and Process File Simultaneously** parameter.

The **C1-FTRAN** batch executes both upload and process in a single execution.



Main

File Request Type *	TXNADD	Description *	transaction upload staging csv
Data Transformation Required	✓	File Atomicity	✓
File Format *	Comma Separated Values	File Extension	csv
Upload and Process File Simultaneously	✓		
File Header Required	✓		
File Footer Required	☐		
Service Log Required	☐	Skip Duplicates	☐
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	3	Validate Record Payload	☐
Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	☐

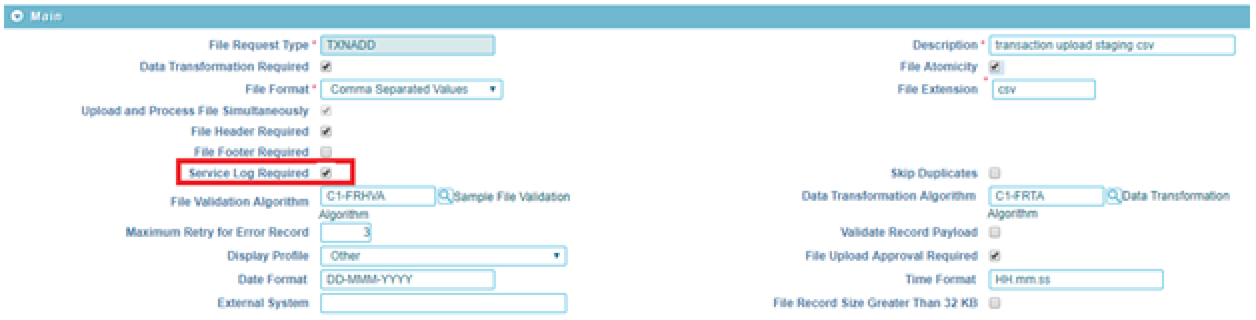
Figure 12: Rollback Processed Records

Note: Since, the batch is executed in single thread, it has a performance impact. Hence, File Atomicity should be opted only in case of low data volume.

Note: **C1-FTRAN** batch is executed using Single Transaction Strategy.

6.1.3 Validating Processing Details

The **Service Log Required** attribute validates if the processing details need to be captured for individual records or not. If you require service log, select **Service Log Required** check box against the File Request Type.

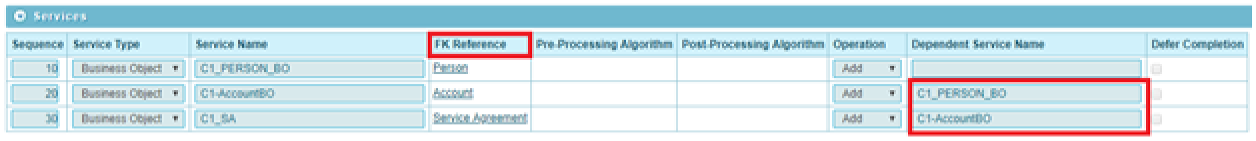


Main

File Request Type *	TXNADD	Description *	transaction upload staging csv
Data Transformation Required	✓	File Atomicity	✓
File Format *	Comma Separated Values	File Extension	csv
Upload and Process File Simultaneously	✓		
File Header Required	✓		
File Footer Required	☐		
Service Log Required	✓	Skip Duplicates	☐
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	3	Validate Record Payload	☐
Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	☐

Figure 13: Service Log

Record details can have a primary key with its service name. Primary key for that record is stamped only if FK reference is configured in the File Request Type for the invoked service.



Services

Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Post-Processing Algorithm	Operation	Dependent Service Name	Defer Completion
10	Business Object	C1_PERSON_BO	Person			Add		☐
20	Business Object	C1-AccountBO	Account			Add	C1_PERSON_BO	☐
30	Business Object	C1_SA	Service Agreement			Add	C1-AccountBO	☐

Figure 14: Service Log and Foreign Key Reference

These details in combination with foreign key reference is used to navigate to its respective entity like 'Person' or 'Account' or 'Contract', etc.

Note: This feature is Optional to optimize the performance of batch.

6.1.4 Header and Footer Details

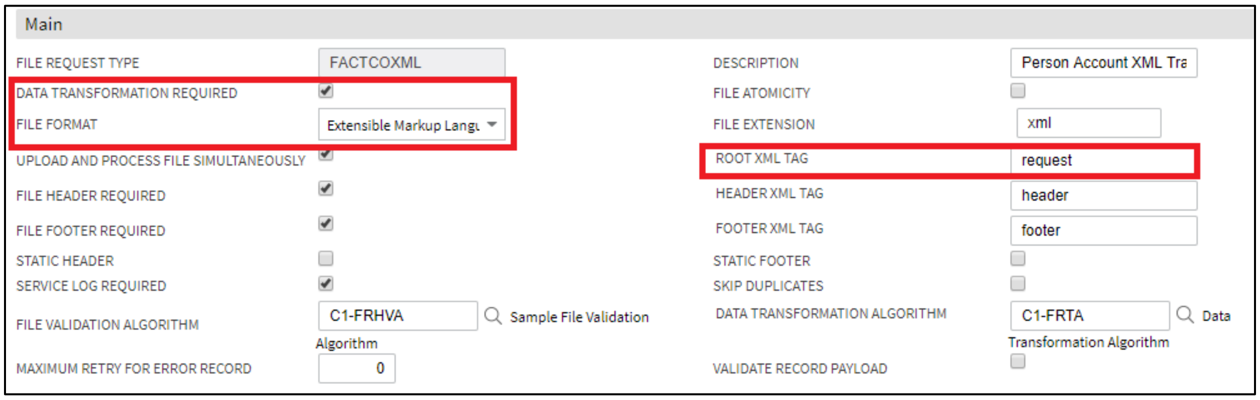
Header and footer details are optional in a file. You can have both header and footer details in a file that you are uploading. If you want to upload a file with header and footer details, you need to configure Header and Footer details in the corresponding File Request Type.

Note: Header and footer details are used for file validations. Header input field value can be referred in Field Transformation using `HEADER(HDR_FLD_NAME)` function.

HEADER – First line record in file is considered as HEADER

FOOTER – Last line record in file is considered as FOOTER

- If **Data Transformation Required** check box is selected and the selected **File Format** is Extensible Markup Language, then:
  - **Root XML Tag** is mandatory. This means the service payload has root XML tag as **<request>……..</request>**

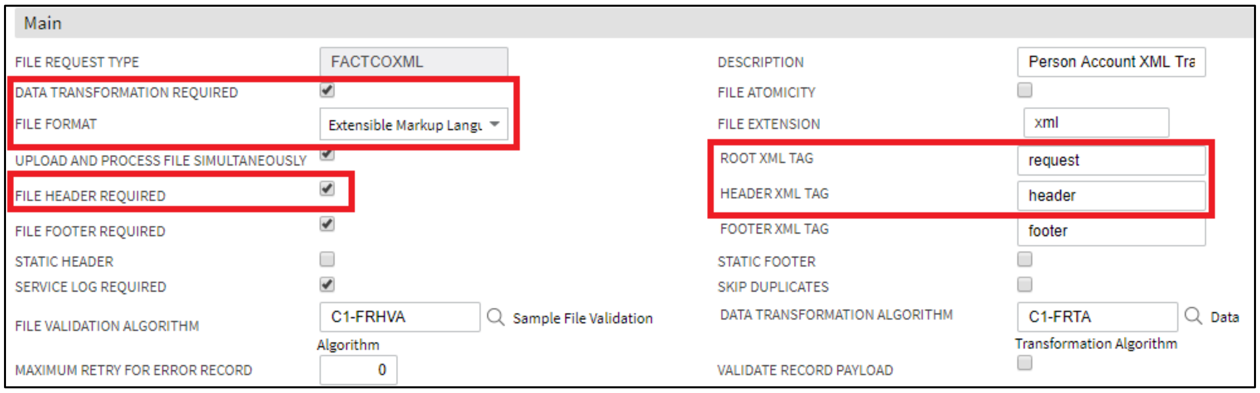


Main

FILE REQUEST TYPE	FACTCOXML	DESCRIPTION	Person Account XML Tra
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	☐
FILE FORMAT	Extensible Markup Langu	FILE EXTENSION	xml
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓	ROOT XML TAG	request
FILE HEADER REQUIRED	✓	HEADER XML TAG	header
FILE FOOTER REQUIRED	✓	FOOTER XML TAG	footer
STATIC HEADER	☐	STATIC FOOTER	☐
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	☐
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	0	VALIDATE RECORD PAYLOAD	☐

Figure 15: Root XML Tag

  - If **File Header Required** check box is selected, **Header XML Tag** is mandatory. This means service payload has header XML tag as **<request> <header>……</header>……..</request>**

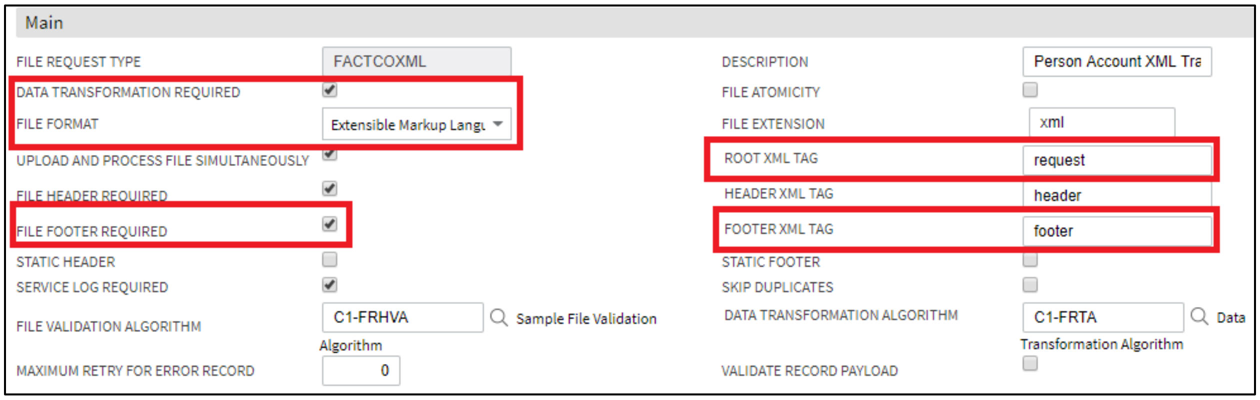


Main

FILE REQUEST TYPE	FACTCOXML	DESCRIPTION	Person Account XML Tra
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	☐
FILE FORMAT	Extensible Markup Langu	FILE EXTENSION	xml
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓	ROOT XML TAG	request
FILE HEADER REQUIRED	✓	HEADER XML TAG	header
FILE FOOTER REQUIRED	✓	FOOTER XML TAG	footer
STATIC HEADER	☐	STATIC FOOTER	☐
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	☐
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	0	VALIDATE RECORD PAYLOAD	☐

Figure 16: Header XML Tag

  - If **File Footer Required** check box is selected, **Footer XML Tag** is mandatory. This means service payload has header XML tag as **<request>……..<footer>…..</footer></request>**

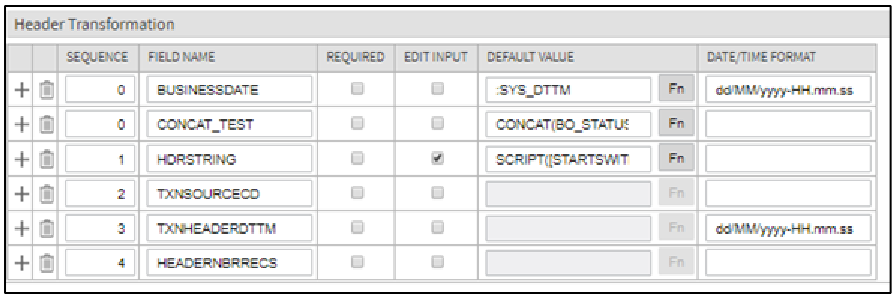


Main

FILE REQUEST TYPE	FACTCOXML	DESCRIPTION	Person Account XML Tra
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	☐
FILE FORMAT	Extensible Markup Langu	FILE EXTENSION	xml
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓	ROOT XML TAG	request
FILE HEADER REQUIRED	✓	HEADER XML TAG	header
FILE FOOTER REQUIRED	✓	FOOTER XML TAG	footer
STATIC HEADER	☐	STATIC FOOTER	☐
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	☐
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	0	VALIDATE RECORD PAYLOAD	☐

Figure 17: Footer XML Tag

- If **Data Transformation Required** check box is selected,
  - If **File Header Required** check box is selected, **Header Transformation** should be configured.

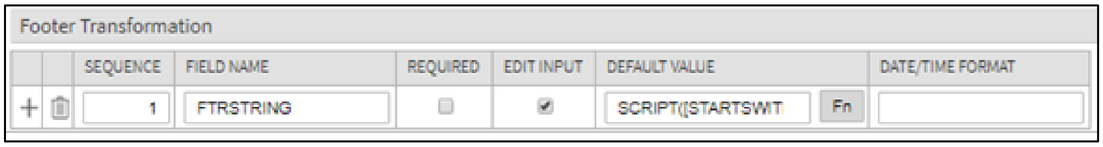


Header Transformation

SEQUENCE	FIELD NAME	REQUIRED	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
0	BUSINESSDATE	☐	☐	:SYS_DTTM	dd/MM/yyyy-HH.mm.ss
0	CONCAT_TEST	☐	☐	CONCAT(BO_STATUS	
1	HDRSTRING	☐	✓	SCRIPT([STARTSWIT	
2	TXNSOURCECD	☐	☐		
3	TXNHEADERDTTM	☐	☐		dd/MM/yyyy-HH.mm.ss
4	HEADERNBRRECS	☐	☐		

Figure 18: Data Transformation - File Header

  - If **File Footer Required** check box is selected, **Footer Transformation** should be configured.



Footer Transformation

SEQUENCE	FIELD NAME	REQUIRED	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
1	FTRSTRING	☐	✓	SCRIPT([STARTSWIT	

Figure 19: Data Transformation – File Footer

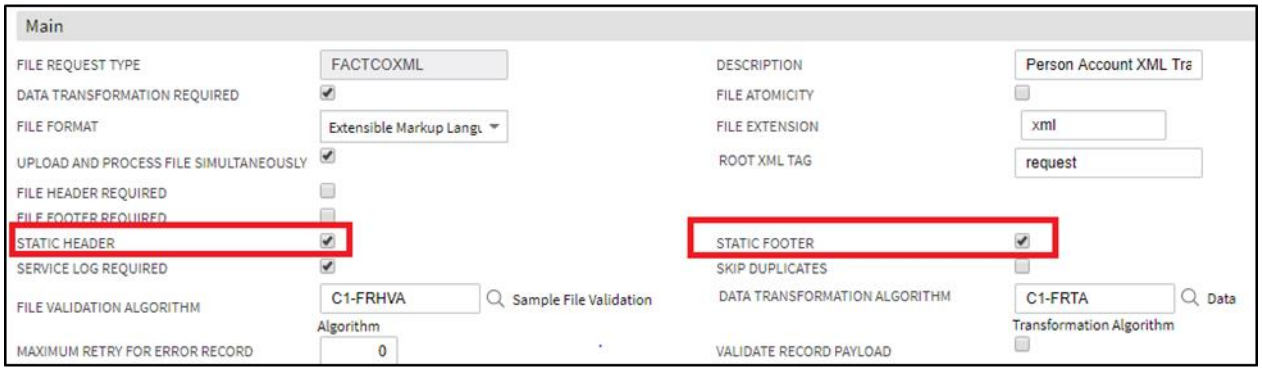
You can also view file header and footer details for files with **Complete** or **Pending** status using the File Upload Dashboard. For information on how to view the file header and footer details, refer to Viewing File Header and Footer Details section in File Upload Interface User Guide.

Static Header and Footer Details

Static Header and Footer indicates that a file does not have any header or footer data line.

- **Static Header** refers to the configuration data done only under Header Transformation.
- **Static Footer** refers to the configuration data done only under Footer Transformation.

These attributes can be used if you want to have any validations done before formation of threads in a batch execution.



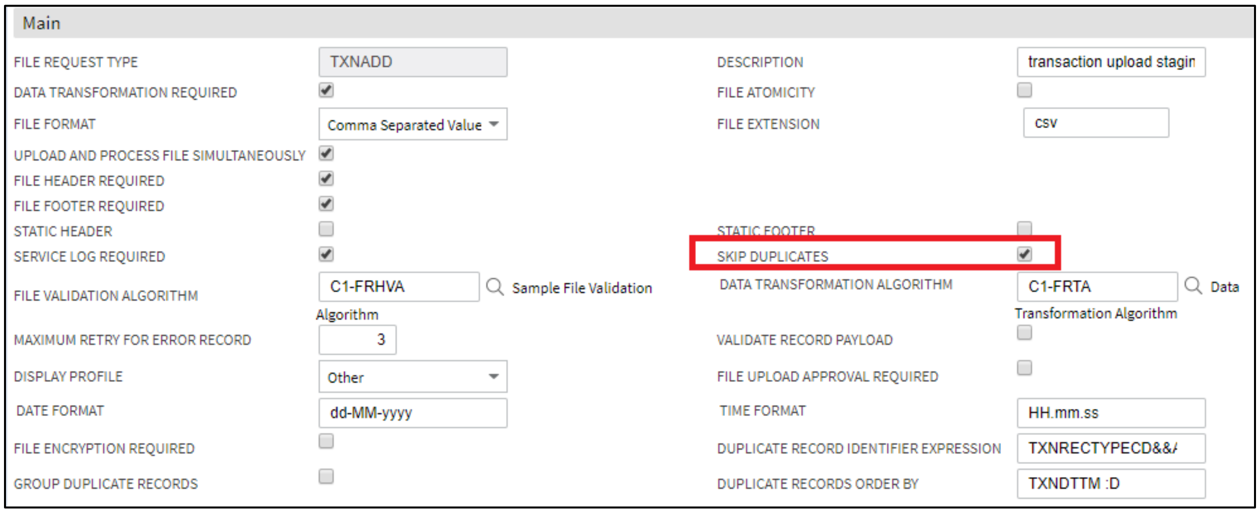
Main

FILE REQUEST TYPE	FACTCOXML	DESCRIPTION	Person Account XML Tra
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	
FILE FORMAT	Extensible Markup Langu	FILE EXTENSION	xml
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓	ROOT XML TAG	request
FILE HEADER REQUIRED			
FILE FOOTER REQUIRED			
STATIC HEADER	✓	STATIC FOOTER	✓
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	0	VALIDATE RECORD PAYLOAD	

Note: You can select either of attributes **STATIC HEADER** or **FILE HEADER**. Same applies for **FOOTER**.

6.1.5 Ignoring or Skipping Duplicate Records in a File

To ignore or skip duplicate records in a file, select **Skip Duplicates** check box against a File Request Type. This sets the flag as True.



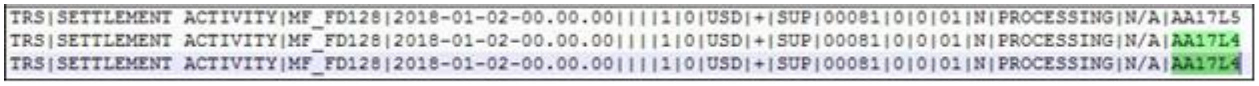
Main

FILE REQUEST TYPE	TXNADD	DESCRIPTION	transaction upload stagin
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	
FILE FORMAT	Comma Separated Value	FILE EXTENSION	csv
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓		
FILE HEADER REQUIRED	✓		
FILE FOOTER REQUIRED	✓		
STATIC HEADER		STATIC FOOTER	
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	✓
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	3	VALIDATE RECORD PAYLOAD	
DISPLAY PROFILE	Other	FILE UPLOAD APPROVAL REQUIRED	
DATE FORMAT	dd-MM-yyyy	TIME FORMAT	HH.mm.ss
FILE ENCRYPTION REQUIRED		DUPLICATE RECORD IDENTIFIER EXPRESSION	TXNRECTYPECD&&/
GROUP DUPLICATE RECORDS		DUPLICATE RECORDS ORDER BY	TXNDTTM :D

Figure 20: Ignore or Skip Duplicate Records

Duplicate records can be categorized into:

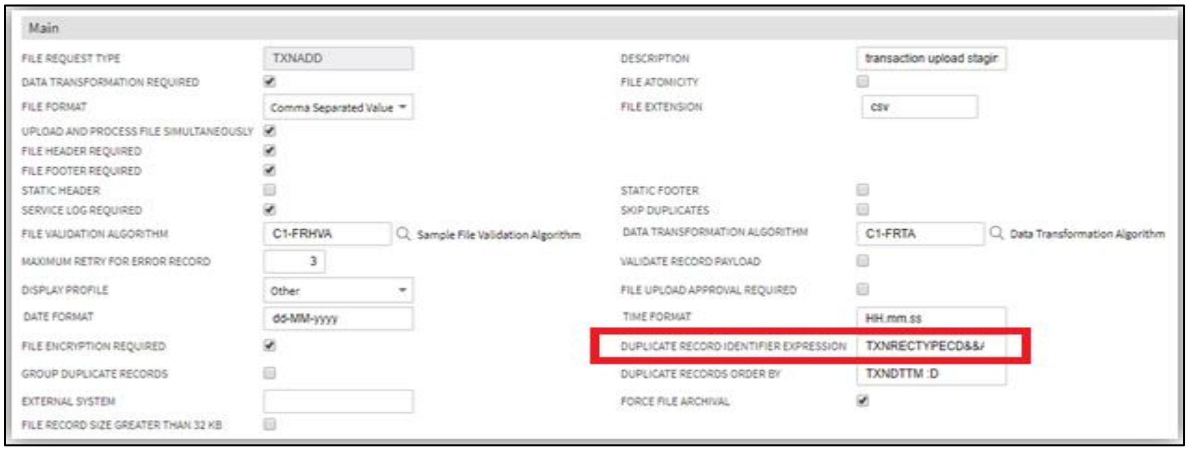
- Duplicate record within the same file. A sample file with duplicate records is represented below:



```
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L5
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L4
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L4
```

In this case, the duplicate records are uploaded with **Ignore** status (marked as **IGN**), and are not processed.

Note: This duplicity is identified on the basis of configured **Duplicate Record Identifier Expression** in Main section of File Request Type.

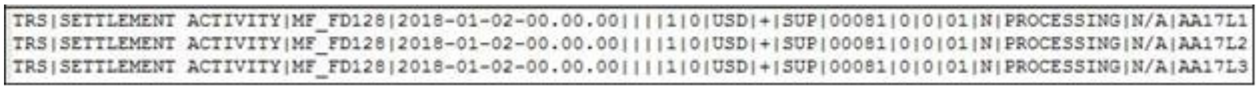


Main

FILE REQUEST TYPE	TXNADD	DESCRIPTION	transaction upload stagir
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	☐
FILE FORMAT	Comma Separated Value	FILE EXTENSION	csv
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓		
FILE HEADER REQUIRED	✓		
FILE FOOTER REQUIRED	✓		
STATIC HEADER	☐	STATIC FOOTER	☐
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	☐
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	3	VALIDATE RECORD PAYLOAD	☐
DISPLAY PROFILE	Other	FILE UPLOAD APPROVAL REQUIRED	☐
DATE FORMAT	dd-MM-yyyy	TIME FORMAT	HH.mm.ss
FILE ENCRYPTION REQUIRED	✓	DUPLICATE RECORD IDENTIFIER EXPRESSION	TXNRECTYPECD&&/
GROUP DUPLICATE RECORDS	☐	DUPLICATE RECORDS ORDER BY	TXNDTTM :D
EXTERNAL SYSTEM		FORCE FILE ARCHIVAL	✓
FILE RECORD SIZE GREATER THAN 32 KB	☐		

Figure 21: Duplicate Record Identifier Expression

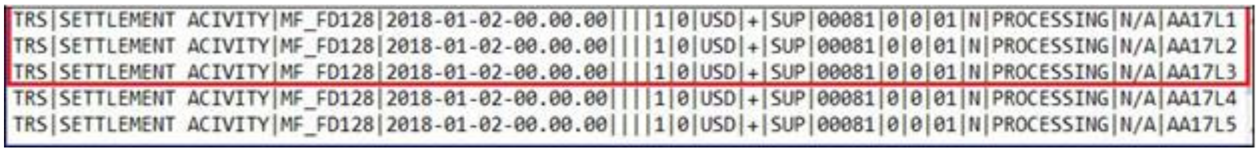
- Records that have been already processed in the previous file upload batch execution. There are two possible scenarios:
    - A text file has three records and is processed using file upload batch. The records are uploaded in ORMB system and the sample file appears like the image below:



```
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L1
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L2
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L3
```

Figure 22: SampleFile_1.txt

    - A text file has five records, out of which first three records have been already uploaded and processed using ‘SampleFile_1.txt’. The duplicate records are uploaded with “SKIP” status and are skipped while processing. A sample file is represented below:



```
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L1
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L2
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L3
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L4
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L5
```

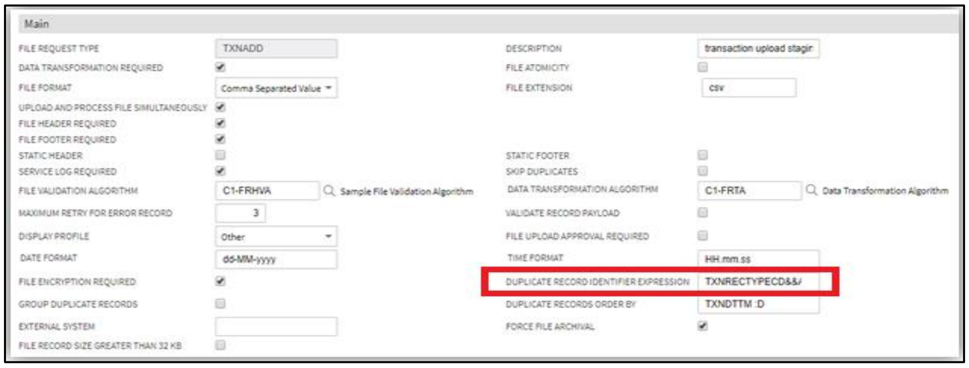
Figure 23: SampleFile_2.txt

Note: This duplicity is identified on the basis of record existence in `CI_FILE_REQUEST_DETAIL` table using “hash key”.

Duplicate Records Identifier Expression

You can have an expression to handle the identical records in the same file based on some file input identifiers. This can be done using Duplicate Records Identifier Expression field in File Request Type page. To form an expression you can use the defined fields in FIELD TRANSFORMATION with a combination of logical operators && and ||.

To support multiple conditions in an expression, you can use **&&** (AND) and **||** (OR) logical operators. The && and || operators perform **Conditional-AND** and **Conditional-OR** operations on two Boolean expressions. These operators exhibit "short-circuiting" behavior, which means that the second operand is evaluated only if needed.



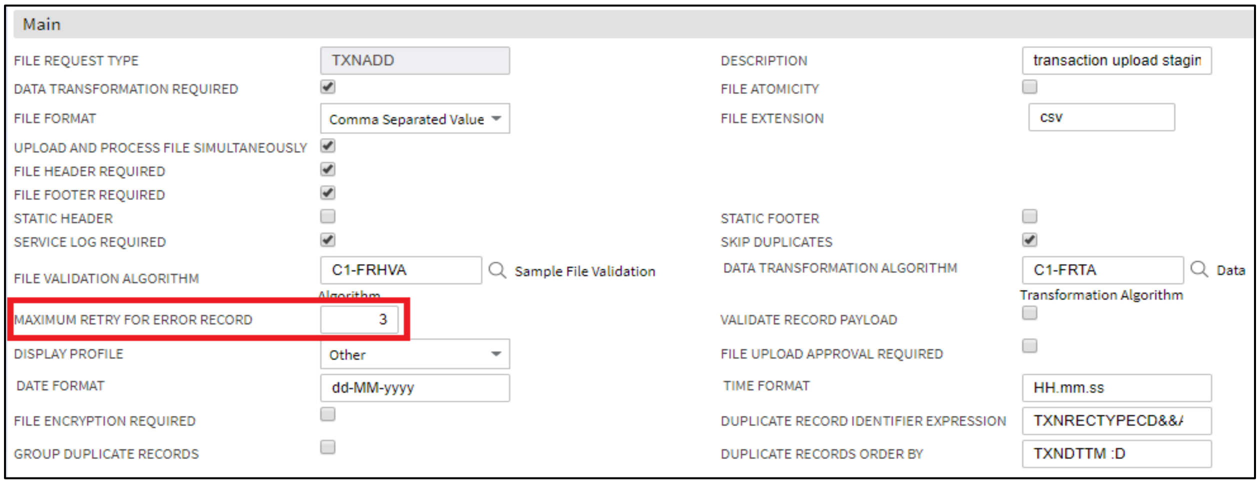
Main

FILE REQUEST TYPE	TXNADD	DESCRIPTION	transaction upload stagir
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	☐
FILE FORMAT	Comma Separated Value	FILE EXTENSION	csv
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓		
FILE HEADER REQUIRED	✓		
FILE FOOTER REQUIRED	✓		
STATIC HEADER	☐	STATIC FOOTER	☐
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	☐
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	3	VALIDATE RECORD PAYLOAD	☐
DISPLAY PROFILE	Other	FILE UPLOAD APPROVAL REQUIRED	☐
DATE FORMAT	dd-MM-yyyy	TIME FORMAT	HH.mm.ss
FILE ENCRYPTION REQUIRED	✓	DUPLICATE RECORD IDENTIFIER EXPRESSION	TXNRECTYPECD&&/
GROUP DUPLICATE RECORDS	☐	DUPLICATE RECORDS ORDER BY	TXNDTTM :D
EXTERNAL SYSTEM		FORCE FILE ARCHIVAL	✓
FILE RECORD SIZE GREATER THAN 32 KB	☐		

Figure 24: Duplicate Record Identifier Expression

6.1.6 Marking the Default Status of Failed Records to "Retry" for Reprocessing

You can configure a value for **Maximum Retry for Error Record** in Main section of File Request Type window. The **Maximum Retry for Error Record** field defines the number of attempts allowed to reprocess a failed record. The value should be greater than zero.

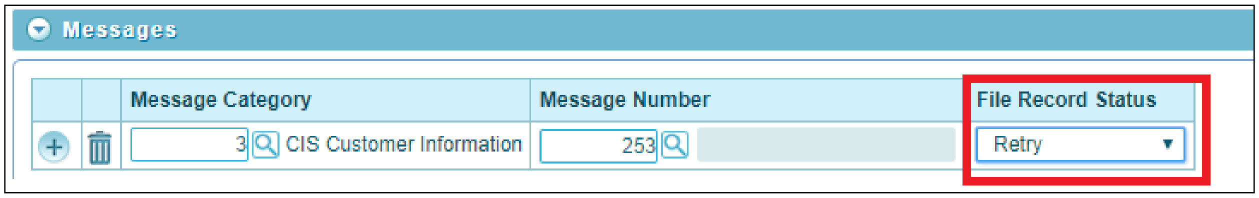


Main

FILE REQUEST TYPE	TXNADD	DESCRIPTION	transaction upload stagin
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	☐
FILE FORMAT	Comma Separated Value	FILE EXTENSION	csv
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓		
FILE HEADER REQUIRED	✓		
FILE FOOTER REQUIRED	✓		
STATIC HEADER	☐	STATIC FOOTER	☐
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	✓
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	3	VALIDATE RECORD PAYLOAD	☐
DISPLAY PROFILE	Other	FILE UPLOAD APPROVAL REQUIRED	☐
DATE FORMAT	dd-MM-yyyy	TIME FORMAT	HH.mm.ss
FILE ENCRYPTION REQUIRED	☐	DUPLICATE RECORD IDENTIFIER EXPRESSION	TXNRECTYPECD&&/
GROUP DUPLICATE RECORDS	☐	DUPLICATE RECORDS ORDER BY	TXNDTTM :D

Figure 25: Maximum Retry for Error Record

To configure the error messages for which the failed records should be marked with Retry status, select Retry option in **File Record Status** drop-down list in the Messages group box.



Messages

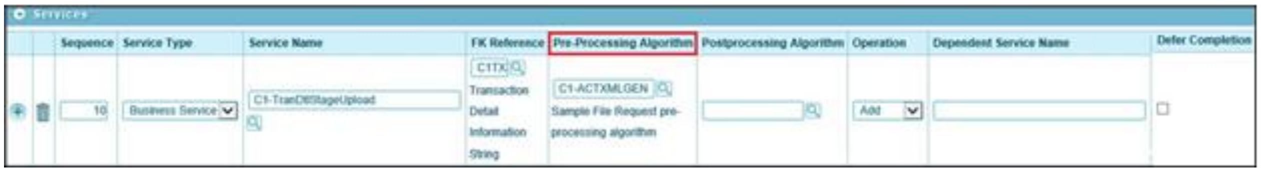
Message Category		Message Number		File Record Status
3	CIS Customer Information	253		Retry

Figure 26: Update Failed Record Error Message

All failed records that are not configured in Messages section of File Request Type, are marked with **Error** status.

6.1.7 Overriding Service Level Operation

You can override service level operations using pre-processing algorithm. The pre-processing algorithm implements the **setOperation(String operation)** method to override the service level operations.



Services

		Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Postprocessing Algorithm	Operation	Dependent Service Name	Defer Completion
		10	Business Service	C1-TranDtlStageUpload	C1TX Transaction Detail Information String	C1-ACTXMLGEN Sample File Request pre-processing algorithm		Add		☐

Figure 27: Override Service Level

After selecting a value in **Pre-Processing Algorithm** field, select **Add** in the Operation drop-down list to override the service level operation.

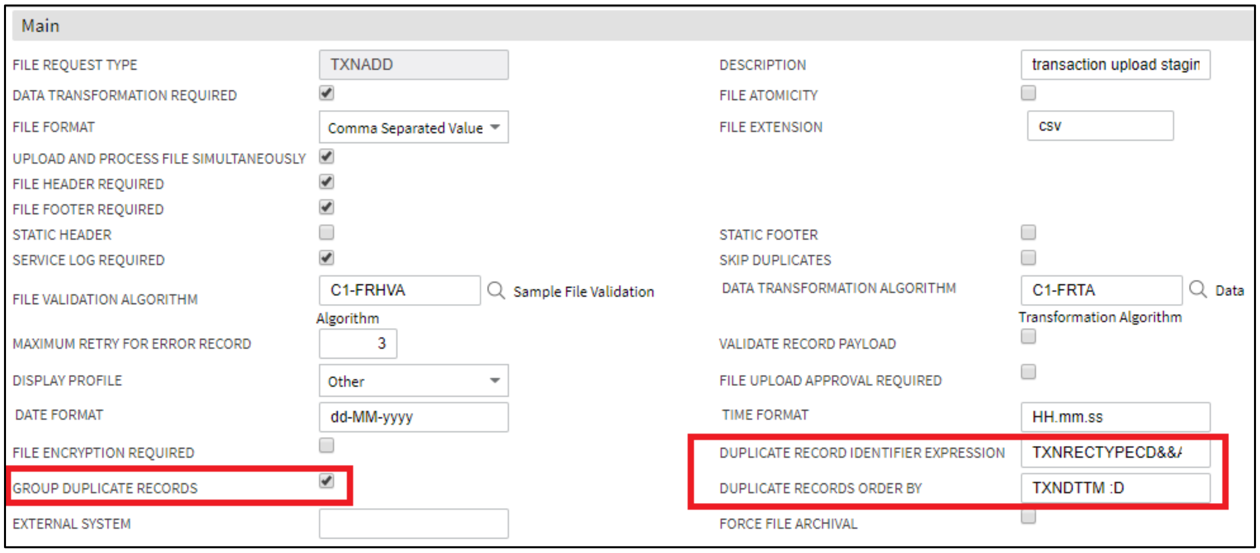
Service level operation can also be overridden using provided FETCH function configuration. Every available FETCH function has the last input parameter SET_OPERATION which is used to reset the configured operation. Set the parameter value to **TRUE** for reset.

To use this FETCH function for operation reset, it has to be 'UPDATE/REPALCE' operation configured in FRT for any service. Real time operation only updates for ADD operation i.e., it will check whether this record exists in ORMB system and if not, reset operation to ADD else execute the service with that operation configured on FRT.

6.1.8 Group Duplicate Records

You can have a group of identical records i.e. records in a same file with same identifiers i.e. identified using Duplicate Record Identifier Expression attribute value. To do this, select **Group Duplicate Records** checkbox.

These group of records are executed in a synchronization as a single work unit. You can define the order by sequence as value of **Duplicate Records Order By** attribute, which allows us to execute the group of records in a sorted order. There can be maximum 2 fields used to define the order by sequence. Default ordering for the fields is ASCENDING. If you want to have the order in DESCENDING then use **:D** with the field name.



Main

FILE REQUEST TYPE	TXNADD	DESCRIPTION	transaction upload stagin
DATA TRANSFORMATION REQUIRED	✓	FILE ATOMICITY	☐
FILE FORMAT	Comma Separated Value	FILE EXTENSION	csv
UPLOAD AND PROCESS FILE SIMULTANEOUSLY	✓		
FILE HEADER REQUIRED	✓		
FILE FOOTER REQUIRED	✓		
STATIC HEADER	☐	STATIC FOOTER	☐
SERVICE LOG REQUIRED	✓	SKIP DUPLICATES	☐
FILE VALIDATION ALGORITHM	C1-FRHVA Sample File Validation Algorithm	DATA TRANSFORMATION ALGORITHM	C1-FRTA Data Transformation Algorithm
MAXIMUM RETRY FOR ERROR RECORD	3	VALIDATE RECORD PAYLOAD	☐
DISPLAY PROFILE	Other	FILE UPLOAD APPROVAL REQUIRED	☐
DATE FORMAT	dd-MM-yyyy	TIME FORMAT	HH.mm.ss
FILE ENCRYPTION REQUIRED	☐	DUPLICATE RECORD IDENTIFIER EXPRESSION	TXNRECTYPECD&&/
GROUP DUPLICATE RECORDS	✓	DUPLICATE RECORDS ORDER BY	TXNDTTM :D
EXTERNAL SYSTEM		FORCE FILE ARCHIVAL	☐

6.1.9 Skipping Service Execution

You can skip service execution for a particular record using a Pre-Processing Algorithm. The algorithm implements the **isSkipServiceExecution()** method and return “true”.

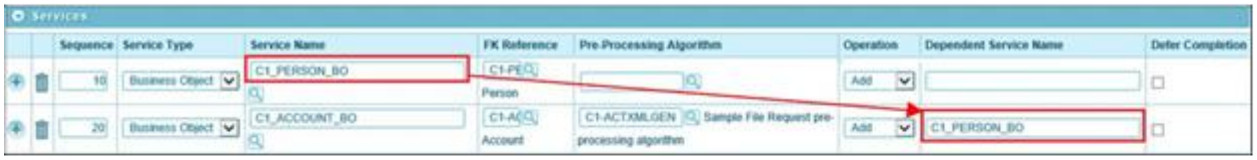
6.1.10 Executing Dependent Service

The **Dependent Service Name** field allows you to address the Payload nesting level and dependent service execution for two independent entities defined in ORMB structure. This signifies a parent-child relationship.

For example, ORMB has two independent entities, **Person** & **Account**, with each entity having their own Maintenance Object (MO). These entities further have Business Object (BO) derived from the corresponding MO. The Business Object has its own XML schema that is specific to the entity.

The legacy system allows you to generate a file with record XML having **Account** nested in **Person**. To configure the service, perform following steps:

1. Click **Add** to add a new sequence.
2. Enter sequence ID in **Sequence** field.
3. Select Business Object from the drop-down list in **Service Type** field.
4. Enter **Service Name** for which you need to configure dependent service name. For example, C1_ACCOUNT_BO
5. Specify **FK Reference** and **Pre-Processing Algorithm**.
6. Enter **Dependent Service Name**. For example, C1_PERSON_BO.



Services

Sequence	Service Type	Service Name	FK Reference	Pre Processing Algorithm	Operation	Dependent Service Name	Defer Completion
10	Business Object	C1_PERSON_BO	C1-PE Person		Add		☐
20	Business Object	C1_ACCOUNT_BO	C1-AC Account	C1-ACTXMLGEN Sample File Request pre-processing algorithm	Add	C1_PERSON_BO	☐

Figure 28: Dependent Service

Dependent service name configuration helps to get the parent’s (Person) service Primary key as an input to its child’s (Account) service pre-processing algorithm.

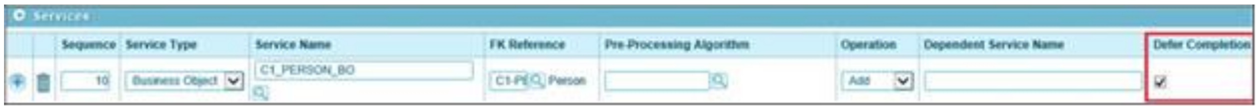
6.1.11 Deferring Completion of Processed Request

Use this flag to defer the completion of successfully processed request. The final status is updated once the corresponding Business Object Life cycle is completed.

It can be used for those Business Objects that have a predefined lifecycle. If a service has **Defer Completion** check box selected, then all the successfully processed records for this service are updated with ‘In-Progress’ status.

Note: Defer Completion is used only to update record status to legacy system.

With this status information, a legacy system can decide an action on the other dependent data upload. For example, if ‘Person’ Business Object has a lifecycle then, corresponding ‘Account’ details is not uploaded until there is an update of ‘Person’ Business Object lifecycle completion.


Services

		Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Operation	Dependent Service Name	Defer Completion
		10	Business Object	C1_PERSON_BO	C1-PE Person		Add		✓

Figure 29: Defer Completion

6.1.12 Supporting Client Defined Date Format

File Upload Interface supports client-defined dates while uploading and transforming file records. You can do this by referring to the Display profile ID at the time of defining or editing or making a copy of a file request type.

Prerequisites

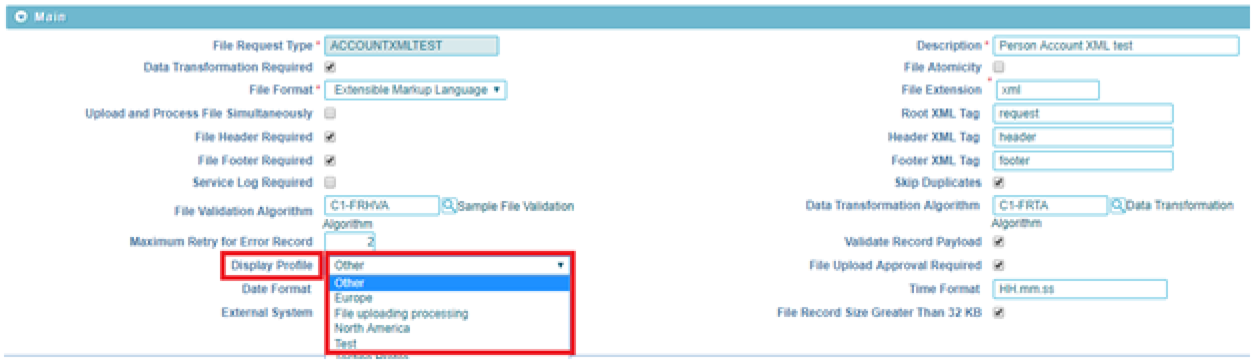
To link display profile ID with a file request type, you should have a Display Profile defined in the application.

Procedure

To link display profile id with file request type, you need to:

1. Enter File Request Type name.
2. Select required fields.
3. Select value from the Display Profile drop-down list.

Note: The file request should have the same date or time format as specified while defining a display profile. An individual File Request Type can support only a single date format.

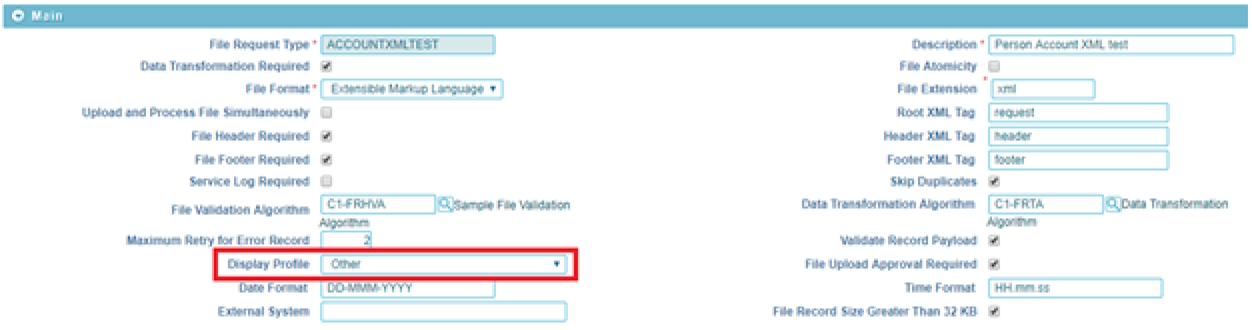

Main

File Request Type *	ACCOUNTXMLTEST	Description *	Person Account XML test
Data Transformation Required	✓	File Atomicity	☐
File Format *	Extensible Markup Language	File Extension *	xml
Upload and Process File Simultaneously	☐	Root XML Tag	request
File Header Required	✓	Header XML Tag	header
File Footer Required	✓	Footer XML Tag	footer
Service Log Required	☐	Skip Duplicates	✓
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	2	Validate Record Payload	✓
Display Profile	Other	File Upload Approval Required	✓
Date Format		Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	✓

Figure 30: Display Profile

6.1.13 Usage of Other Display Profile

Not all Date formats can be configured using Display Profile. To overcome this problem, use the fields 'Date Format' and 'Time Format' on File Request Type. Both these fields are visible only if you select **Other** as the Display Profile.

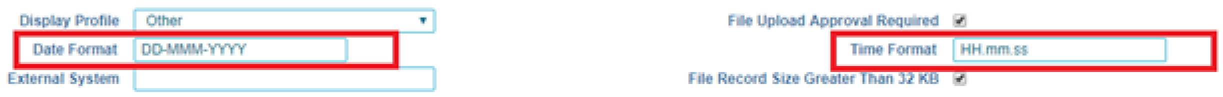


Main

File Request Type *	ACCOUNTXMLTEST	Description *	Person Account XML test
Data Transformation Required	✓	File Atomicity	☐
File Format *	Extensible Markup Language	File Extension *	xml
Upload and Process File Simultaneously	☐	Root XML Tag	request
File Header Required	✓	Header XML Tag	header
File Footer Required	✓	Footer XML Tag	footer
Service Log Required	☐	Skip Duplicates	✓
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	2	Validate Record Payload	✓
Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	✓

Figure 31: Display Profile selection

These field values will have the client supported Date and Time format. Both are mandatory.



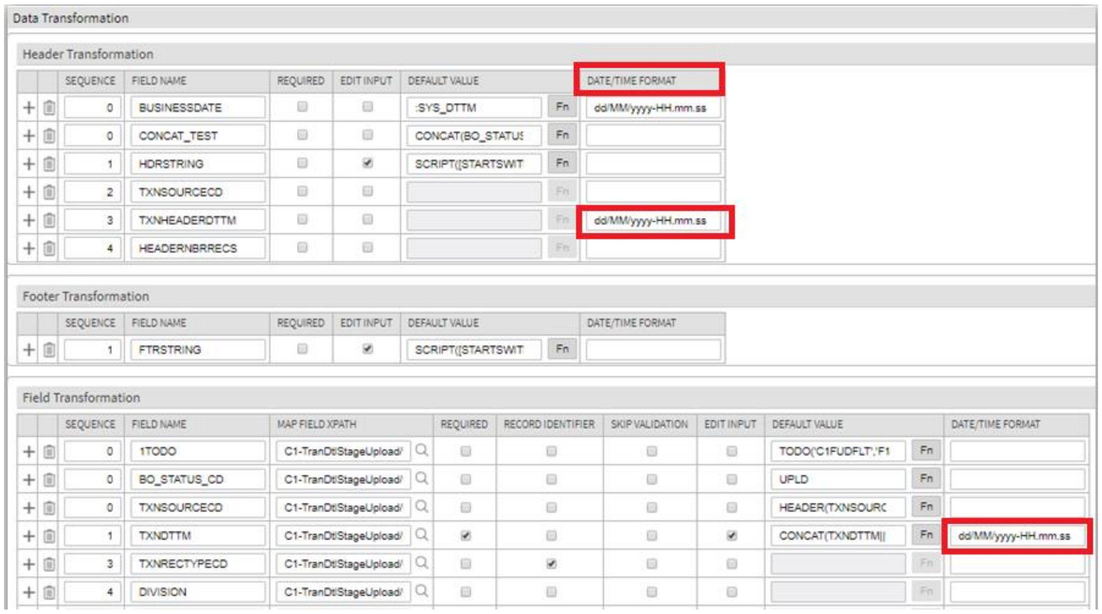
Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	✓

Figure 32: Date Format and Time Format

While processing files with these date formats, they are converted to the framework supported date and time format.

6.1.14 Supporting multiple date formats

You can configure multiple date formats for the required fields in Data Transformation section. Date format can be configured for every required field in the transformation.



Data Transformation

Header Transformation

SEQUENCE	FIELD NAME	REQUIRED	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
0	BUSINESSDATE	☐	☐	:SYS_DTTM	dd/MM/yyyy-HH.mm.ss
0	CONCAT_TEST	☐	☐	CONCAT(BO_STATUS	
1	HDRSTRING	☐	✓	SCRIPT([STARTSWIT	
2	TXNSOURCECD	☐	☐		
3	TXNHEADERDTTM	☐	☐		dd/MM/yyyy-HH.mm.ss
4	HEADERNBRRECS	☐	☐		

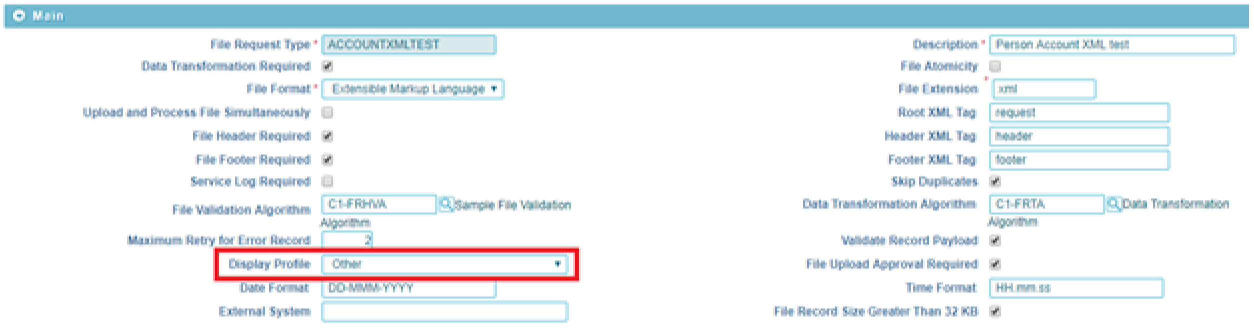
Footer Transformation

SEQUENCE	FIELD NAME	REQUIRED	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
1	FTRSTRING	☐	✓	SCRIPT([STARTSWIT	

Field Transformation

SEQUENCE	FIELD NAME	MAP FIELD XPATH	REQUIRED	RECORD IDENTIFIER	SKIP VALIDATION	EDIT INPUT	DEFAULT VALUE	DATE/TIME FORMAT
0	1TODO	C1-TranDtlStageUpload/	☐	☐	☐	☐	TODO('C1FUDFLT','F1	
0	BO_STATUS_CD	C1-TranDtlStageUpload/	☐	☐	☐	☐	UPLD	
0	TXNSOURCECD	C1-TranDtlStageUpload/	☐	☐	☐	☐	HEADER(TXNSOURC	
1	TXNDTTM	C1-TranDtlStageUpload/	✓	☐	☐	✓	CONCAT(TXNDTTM||	dd/MM/yyyy-HH.mm.ss
3	TXNRECTYPECD	C1-TranDtlStageUpload/	☐	✓	☐	☐		
4	DIVISION	C1-TranDtlStageUpload/	☐	☐	☐	☐		

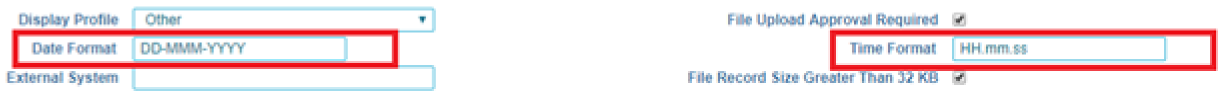
Figure 33: Multiple Date Formats



File Request Type *	ACCOUNTXMLTEST	Description *	Person Account XML test
Data Transformation Required	✓	File Atomicity	☐
File Format *	Extensible Markup Language	File Extension *	xml
Upload and Process File Simultaneously	☐	Root XML Tag	request
File Header Required	✓	Header XML Tag	header
File Footer Required	✓	Footer XML Tag	footer
Service Log Required	☐	Skip Duplicates	✓
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	2	Validate Record Payload	✓
Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	✓

Figure 34: Display Profile selection

These field values have the client supported Date and Time format. Both are mandatory.



Display Profile	Other	File Upload Approval Required	✓
Date Format	DD-MMM-YYYY	Time Format	HH.mm.ss
External System		File Record Size Greater Than 32 KB	✓

Figure 35: Date Format and Time Format

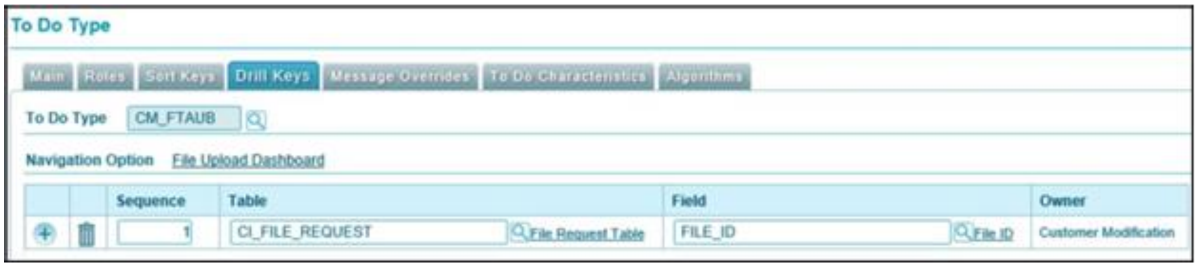
While processing files with these date formats, they are converted to the framework supported date and time format.

6.1.15 Viewing TO DO Entries for Failed Files

If an exception occurs while executing the File Upload and Transformation batch, you can notify the user about the exception by generating a **To Do**. You can view the exception details in To Do Entry, where you see details like the total number of To Do Entries that are open, or are being worked on, or completed against each To Do Type in the file transformation and upload process.

To Do is created using its corresponding To Do Type.

Note: While defining a To Do Type, it is mandatory to set up drill key of a file ID.



To Do Type CM_FTAUB

Navigation Option File Upload Dashboard

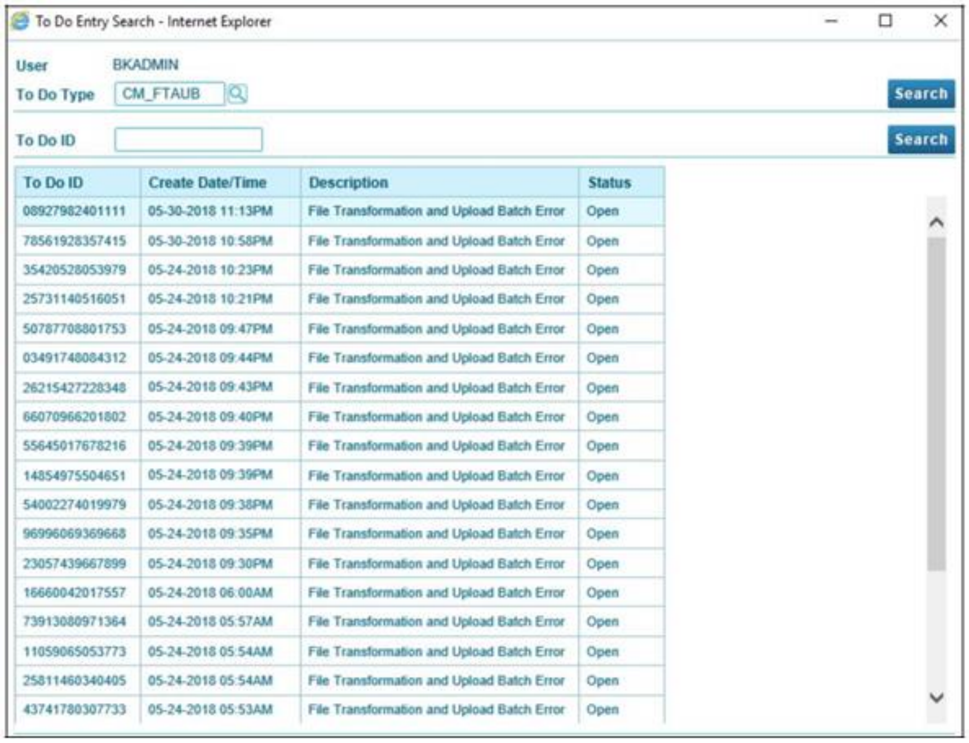
Sequence	Table	Field	Owner
1	CI_FILE_REQUEST File Request Table	FILE_ID File ID	Customer Modification

Figure 36: To Do Type - Drill Keys

Once you create a To Do for capturing File Transformation and Upload Batch Error, then on submitting the respective file request, the file transformation and upload process creates a To Do for the user to review the file request. You can perform various actions or modify details of a To Do entry.

To view To Do entries:

1. From the Admin menu, select To Do and then click To Do Entry. A sub-menu appears.
2. Click **Search** option from the To Do Entry sub-menu. The To Do Entry Search window appears.
3. Enter a To Do Type and click Search. The grid that follows contains the To Do entries that match your criteria.



User BKADMIN

To Do Type CM_FTAUB

To Do ID

To Do ID	Create Date/Time	Description	Status
08927982401111	05-30-2018 11:13PM	File Transformation and Upload Batch Error	Open
78561928357415	05-30-2018 10:58PM	File Transformation and Upload Batch Error	Open
35420528053979	05-24-2018 10:23PM	File Transformation and Upload Batch Error	Open
25731140516051	05-24-2018 10:21PM	File Transformation and Upload Batch Error	Open
50787708801753	05-24-2018 09:47PM	File Transformation and Upload Batch Error	Open
03491748084312	05-24-2018 09:44PM	File Transformation and Upload Batch Error	Open
26215427228348	05-24-2018 09:43PM	File Transformation and Upload Batch Error	Open
66070966201802	05-24-2018 09:40PM	File Transformation and Upload Batch Error	Open
55645017678216	05-24-2018 09:39PM	File Transformation and Upload Batch Error	Open
14854975504651	05-24-2018 09:39PM	File Transformation and Upload Batch Error	Open
54002274019979	05-24-2018 09:38PM	File Transformation and Upload Batch Error	Open
96996069369668	05-24-2018 09:35PM	File Transformation and Upload Batch Error	Open
23057439667899	05-24-2018 09:30PM	File Transformation and Upload Batch Error	Open
16660042017557	05-24-2018 06:00AM	File Transformation and Upload Batch Error	Open
73913080971364	05-24-2018 05:57AM	File Transformation and Upload Batch Error	Open
11059065053773	05-24-2018 05:54AM	File Transformation and Upload Batch Error	Open
25811460340405	05-24-2018 05:54AM	File Transformation and Upload Batch Error	Open
43741780307733	05-24-2018 05:53AM	File Transformation and Upload Batch Error	Open

Figure 37: To Do Entry Search Results Grid

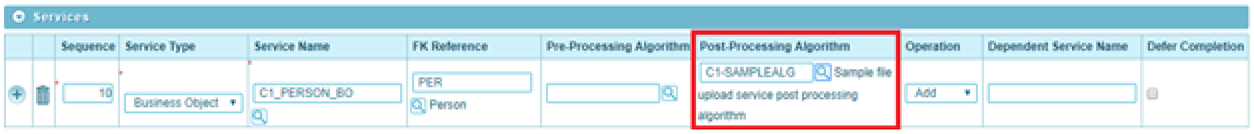
The following information appears:

Column Name	**Description**
To Do ID	Used to indicate the unique identifier of the To Do entry
Create Date/Time	Used to indicate the date and time the To Do entry was created by the system
Description	Used to indicate the description of the To Do Type
Status	Used to indicate the current status of the To Do entry The valid values are: • Open • Being Worked On • Complete

4. Click on text portion in any of the columns to view the respective To Do Details.

6.1.16 Post Processing Algorithm Support for File Request

You can define a post-processing algorithm, which is used to undertake some post-processing activities after successful processing of a record.



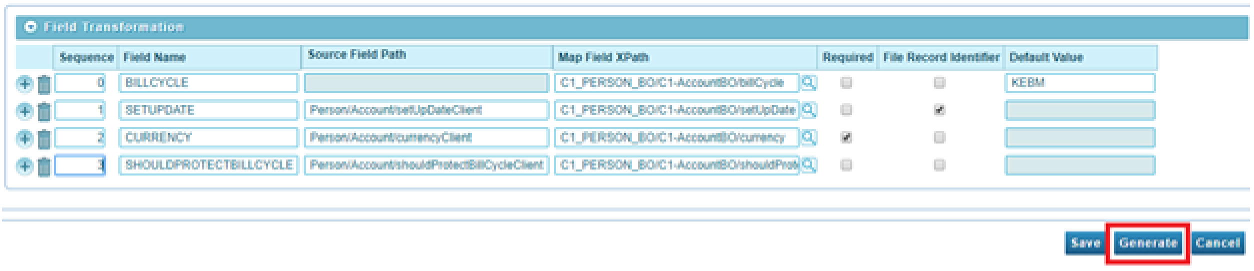
		Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Post-Processing Algorithm	Operation	Dependent Service Name	Defer Completion
		10	Business Object	C1_PERSON_BO	PER Person		C1-SAMPLEALG Sample file upload service post processing algorithm	Add		

Figure 38: Post-Processing Algorithm

To attach a post-processing algorithm to file request type, specify the post-processing algorithm code in Post Processing Algorithm field within Services section.

6.1.17 Auto Generate Field Transformation

Generate button can be used to generate data transformation details for each individual service mentioned in that File Request Type.

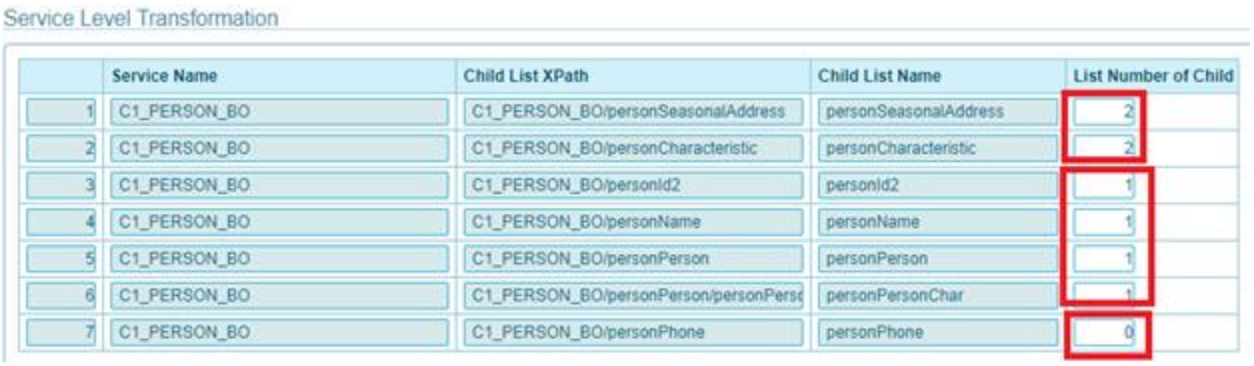


Field Transformation

Sequence	Field Name	Source Field Path	Map Field XPath	Required	File Record Identifier	Default Value
0	BILLCYCLE		C1_PERSON_BO/C1-AccountBO/billCycle	☐	☐	KEBM
1	SETUPDATE	Person/Account/setUpDateClient	C1_PERSON_BO/C1-AccountBO/setUpDate	☐	✓	
2	CURRENCY	Person/Account/currencyClient	C1_PERSON_BO/C1-AccountBO/currency	✓	☐	
3	SHOULDPROTECTBILLCYCLE	Person/Account/shouldProtectBillCycleClient	C1_PERSON_BO/C1-AccountBO/shouldProt	☐	☐	

Figure 39: Auto Generate

You can configure the number of child blocks required in the service schema. It can be zero or more.



Service Level Transformation

	Service Name	Child List XPath	Child List Name	List Number of Child
1	C1_PERSON_BO	C1_PERSON_BO/personSeasonalAddress	personSeasonalAddress	2
2	C1_PERSON_BO	C1_PERSON_BO/personCharacteristic	personCharacteristic	2
3	C1_PERSON_BO	C1_PERSON_BO/personId2	personId2	1
4	C1_PERSON_BO	C1_PERSON_BO/personName	personName	1
5	C1_PERSON_BO	C1_PERSON_BO/personPerson	personPerson	1
6	C1_PERSON_BO	C1_PERSON_BO/personPerson/personPers	personPersonChar	1
7	C1_PERSON_BO	C1_PERSON_BO/personPhone	personPhone	0

Figure 40: Schema level Transformation

Sequence Number for each field is based on the defined order in schema. For fixed position file format, Field, start position and end position will be populated using its defined order in schema and its defined precision i.e. length.

You can edit auto generated data transformation details. For any addition or updates or deletion of field details, data transformation Sequence will be auto reset.

If we don’t want to go with sequence auto reset, then, uncheck the “Transformation Auto Sequence Reset” checkbox under File Request Type. This field on File Request Type will be only visible after auto generation of data transform details.

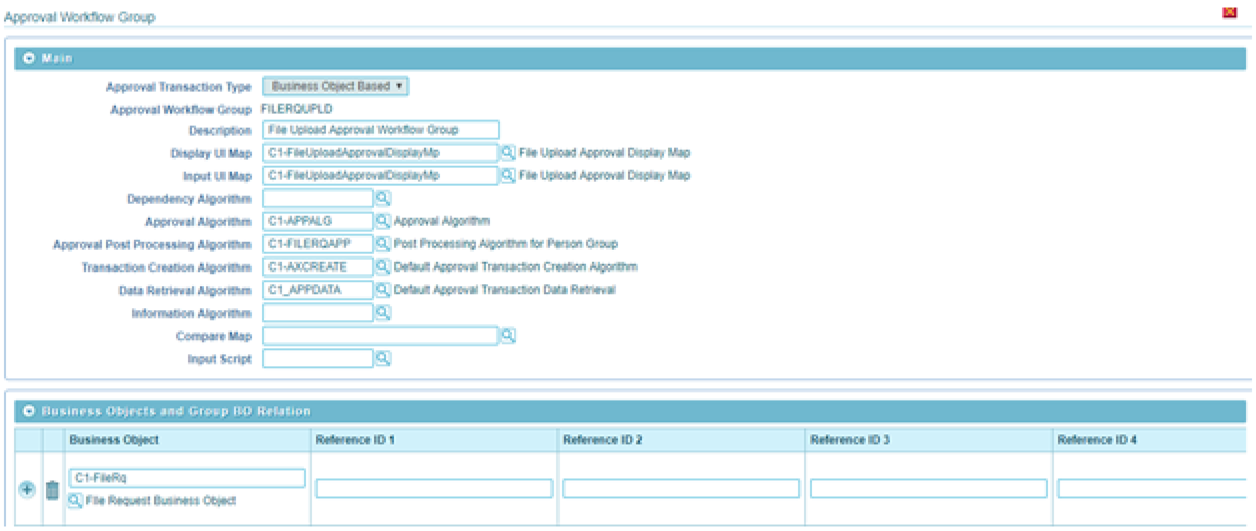
6.2 Approval Workflow configuration for File Upload Interface

Steps for configuration of approval workflow for File Upload Interface:

6.2.1 Create Approval Workflow Group

1. Go to Admin Menu > A > Approval Workflow Group.

2. Click on Add button.



Approval Workflow Group

Main

Approval Transaction Type	Business Object Based
Approval Workflow Group	FILERQUPLD
Description	File Upload Approval Workflow Group
Display UI Map	C1-FileUploadApprovalDisplayMp File Upload Approval Display Map
Input UI Map	C1-FileUploadApprovalDisplayMp File Upload Approval Display Map
Dependency Algorithm	
Approval Algorithm	C1-APPALG Approval Algorithm
Approval Post Processing Algorithm	C1-FILERQAPP Post Processing Algorithm for Person Group
Transaction Creation Algorithm	C1-AXCREATE Default Approval Transaction Creation Algorithm
Data Retrieval Algorithm	C1_APPDATA Default Approval Transaction Data Retrieval
Information Algorithm	
Compare Map	
Input Script	

Business Objects and Group BO Relation

Business Object	Reference ID 1	Reference ID 2	Reference ID 3	Reference ID 4
C1-FileRq File Request Business Object				

Figure 41: Approval Workflow Group

3. Enter the details and click on save button.

   Main group box:

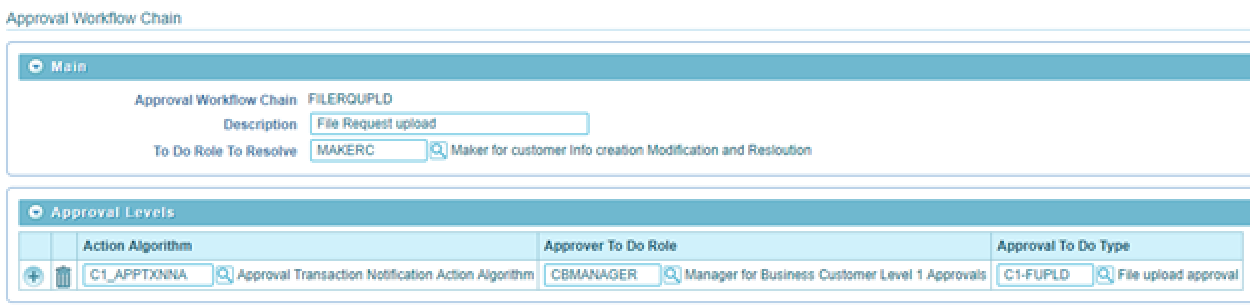
Field Name	Description
Approval Transaction Type	Business Object Based
Approval Workflow Group	FILERQUPLD
Display UI Map	C1-FileUploadApprovalDisplayMp
Input UI Map	C1-FileUploadApprovalDisplayMp
Approval Algorithm	C1-APPALG
Approval Post Processing Algorithm	C1-FILERQAPP
Transaction Creation Algorithm	C1-AXCREATE
Data Retrieval Algorithm	C1_APPDATA

Business Objects and Group BO Relation:

Field Name	Description
Business Object	C1-FileRq
List	FALSE

6.2.2 Create Approval Workflow Chain

1. Go to Admin Menu > A > Approval Workflow Chain and click Add.


Approval Workflow Chain

Main

Approval Workflow Chain	FILERQUPLD
Description	File Request upload
To Do Role To Resolve	MAKERC Maker for customer Info creation Modification and Resloution

Approval Levels

Action Algorithm	Approver To Do Role	Approval To Do Type
C1_APPTXNNA Approval Transaction Notification Action Algorithm	CBMANAGER Manager for Business Customer Level 1 Approvals	C1-FUPLD File upload approval

Figure 42: Approval Workflow Chain

2. Enter the required details and click Save.

Main group box:

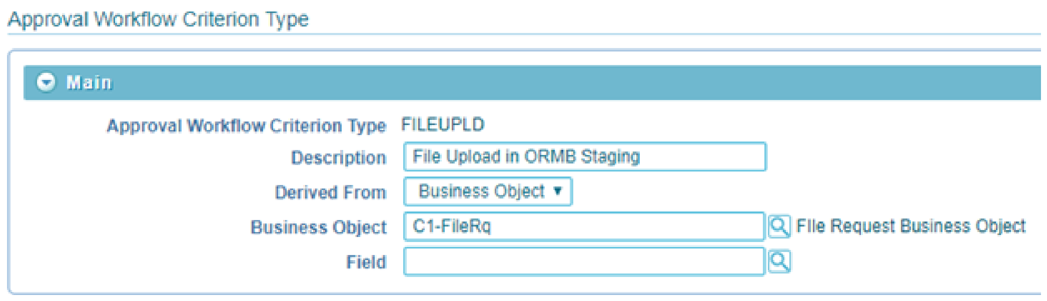
Field Name	Description
Approval Workflow Chain	FILERQUPLD
To Do Role To Resolve	MAKERC

Approval Levels group box:

Field Name	Description
Action Algorithm	C1_APPTXNNA
Approval To Do Type	C1-FUPLD

6.2.3 Create Approval Workflow Criterion Type

1. Go to Admin Menu > A > Approval Workflow Criterion Type and click Add.


Approval Workflow Criterion Type

Main

Approval Workflow Criterion Type	FILEUPLD
Description	File Upload in ORMB Staging
Derived From	Business Object
Business Object	C1-FileRq File Request Business Object
Field	

Figure 43: Approval Workflow Criterion Type

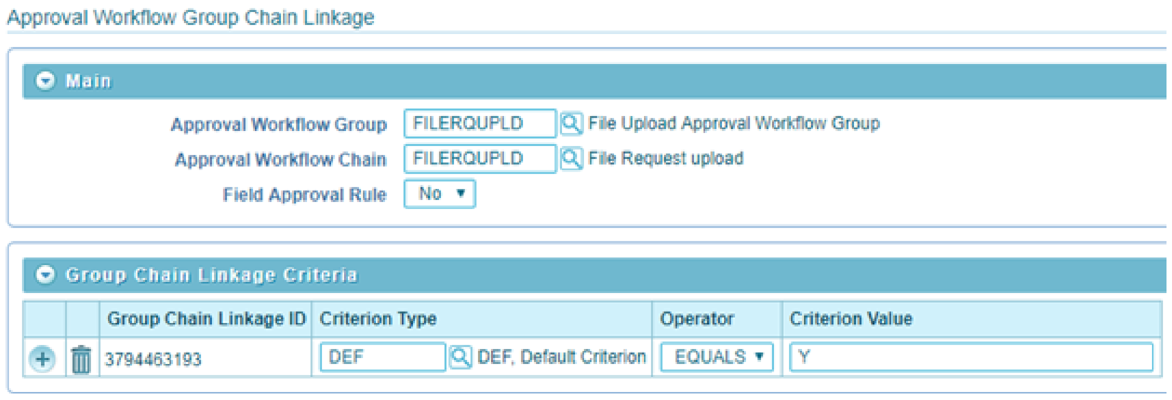
2. Enter the required details and click on Save.

Main group box:

Field Name	Description
Approval Workflow Criterion Type	FILEUPLD
Derived From	Business Object
Business Object	C1-FileRq

6.2.4 Create Approval Workflow Group Chain Linkage

1. Go to Admin Menu > A > Approval Workflow Group Chain Linkage and click Add.
2. Enter the required details and click on Save.


Approval Workflow Group Chain Linkage

Main

Approval Workflow Group: FILERQUPLD File Upload Approval Workflow Group

Approval Workflow Chain: FILERQUPLD File Request upload

Field Approval Rule: No

Group Chain Linkage Criteria

		Group Chain Linkage ID	Criterion Type	Operator	Criterion Value
+	🗑	3794463193	DEF DEF, Default Criterion	EQUALS	Y

Figure 44: Approval Workflow Group Chain Linkage

Main group box:

Field Name	Description
Approval Workflow Group	FILERQUPLD
Approval Workflow Chain	FILERQUPLD
Field Approval Rule	No

Group Chain Linkage Criteria:

Field Name	Description
Criterion Type	DEF
Operator	EQUALS
Criterion Value	Y

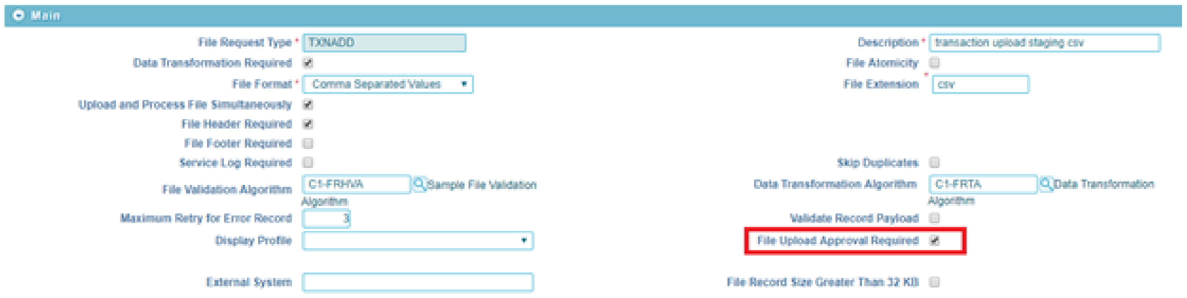
6.2.5 Create Approval Workflow Settings

1. Go to Admin Menu > A > Approval Workflow Settings and click Add.
2. Enter the required details and click Save.

Main group box:

Field Name	Description
Approval Workflow Group	FILERQUPLD
Approval Chain Selection Algorithm	C1-APPCHAIN
Display UI Map	No
Input UI Map	Yes
Approval Algorithm	Yes

6.2.6 Update "File Upload Approval Required" flag as true



Main

File Request Type *	TXNADD	Description *	transaction upload staging csv
Data Transformation Required	✓	File Atomicity	☐
File Format *	Comma Separated Values	File Extension *	csv
Upload and Process File Simultaneously	✓		
File Header Required	✓		
File Footer Required	☐		
Service Log Required	☐	Skip Duplicates	☐
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	3	Validate Record Payload	☐
Display Profile		File Upload Approval Required	✓
External System		File Record Size Greater Than 32 KB	☐

Figure 45: File Upload Approval Required

7. Updating Records marked with 'Error' or 'Pending' status

When updating records marked with 'Error' or 'Pending' status, you have two possible modes:

- Update records marked with 'Error' status to 'Retry' status
- Update records marked with 'Pending' status to 'Error' status

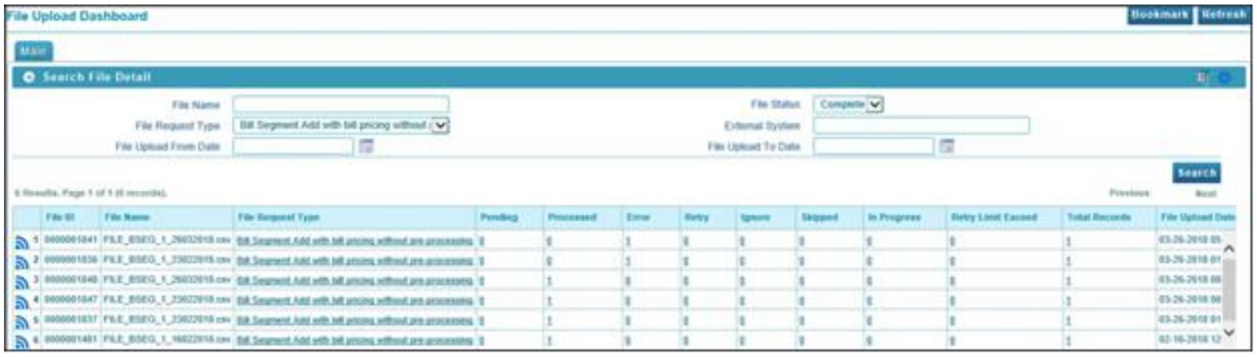
ORMB provides you two options to update records.

- File Upload Dashboard
- File Request Status Update Batch

7.1 File Upload Dashboard

This is used to update status for less number of records. To update records with "Error" to "Retry" status, you need to follow below steps:

1. Click the Menu link in the Application toolbar. A list appears.
2. Select Tools from the list. A sub-menu appears.
3. Click File Upload Dashboard option. The Search File window appears.
4. Enter either File Name or select File Request Type from the drop-down list. Click Search.
5. The file details that meet the search criteria appear in the Search Results section in a tabular format.



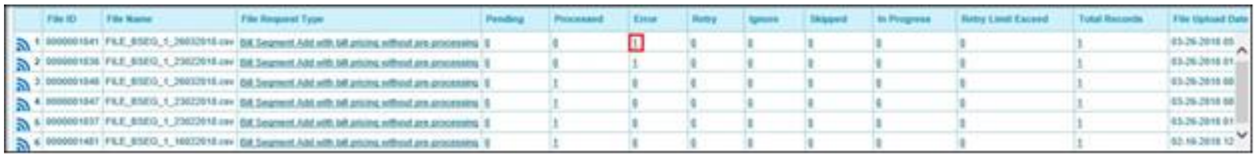
File Name	
File Request Type	Bill Segment Add with bill pricing without
File Upload From Date	
File Status	Complete
External System	
File Upload To Date	

6 Results. Page 1 of 1 (6 records).

	File ID	File Name	File Request Type	Pending	Processed	Error	Retry	Ignore	Skipped	In Progress	Retry Limit Exceed	Total Records	File Upload Date
1	0000001841	FILE_BSEG_1_26032018.csv	Bill Segment Add with bill pricing without pre-processing	0	0	1	0	0	0	0	0	1	03-26-2018 05
2	0000001836	FILE_BSEG_1_23022018.csv	Bill Segment Add with bill pricing without pre-processing	0	0	1	0	0	0	0	0	1	03-26-2018 01
3	0000001840	FILE_BSEG_1_26032018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	03-26-2018 00
4	0000001847	FILE_BSEG_1_23022018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	03-26-2018 00
5	0000001837	FILE_BSEG_1_23022018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	03-26-2018 01
6	0000001481	FILE_BSEG_1_16022018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	02-16-2018 12

Figure 46: File Upload Dashboard

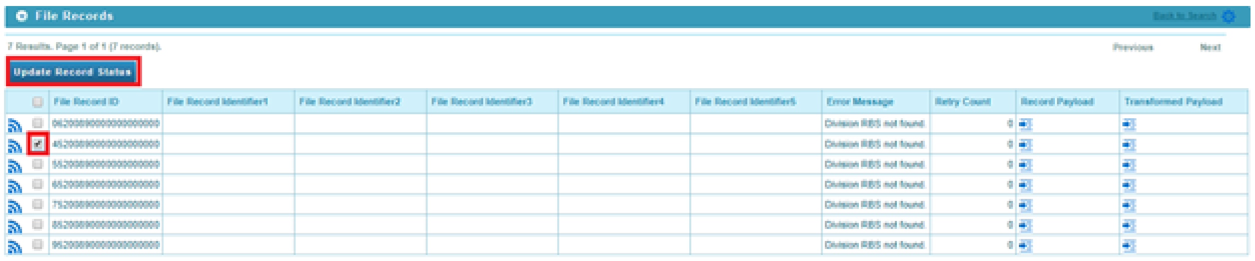
6. The counts in respective Status columns represent the number of records. The numbers are linked to Search File Record Detail form.
7. Click the number in Error column to view the record details.



	File ID	File Name	File Request Type	Pending	Processed	Error	Retry	Ignore	Skipped	In Progress	Retry Limit Exceed	Total Records	File Upload Date
1	0000001841	FILE_BSEG_1_26032018.csv	Bill Segment Add with bill pricing without pre-processing	0	0	1	0	0	0	0	0	1	03-26-2018 05
2	0000001836	FILE_BSEG_1_23022018.csv	Bill Segment Add with bill pricing without pre-processing	0	0	1	0	0	0	0	0	1	03-26-2018 01
3	0000001840	FILE_BSEG_1_26032018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	03-26-2018 00
4	0000001847	FILE_BSEG_1_23022018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	03-26-2018 00
5	0000001837	FILE_BSEG_1_23022018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	03-26-2018 01
6	0000001481	FILE_BSEG_1_16022018.csv	Bill Segment Add with bill pricing without pre-processing	0	1	0	0	0	0	0	0	1	02-16-2018 12

Figure 47: Error Records

8. File Records window appears. Select records for which "Retry" status is to be updated.



File Records

Back to Search

7 Results. Page 1 of 1 (7 records).

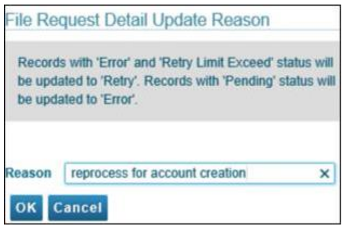
Previous Next

Update Record Status

		File Record ID	File Record Identifier1	File Record Identifier2	File Record Identifier3	File Record Identifier4	File Record Identifier5	Error Message	Retry Count	Record Payload	Transformed Payload
	☐	06200890000000000000						Division RBS not found.	0		
	✓	45200890000000000000						Division RBS not found.	0		
	☐	55200890000000000000						Division RBS not found.	0		
	☐	65200890000000000000						Division RBS not found.	0		
	☐	75200890000000000000						Division RBS not found.	0		
	☐	85200890000000000000						Division RBS not found.	0		
	☐	95200890000000000000						Division RBS not found.	0		

Figure 48: File Records

9. Click Update Record Status. The File Request Detail Update Reason dialog box appears.



File Request Detail Update Reason

Records with 'Error' and 'Retry Limit Exceed' status will be updated to 'Retry'. Records with 'Pending' status will be updated to 'Error'.

Reason: reprocess for account creation

Figure 49: File Request Detail Update Reason

10. Specify the reason and click OK. Similarly, you need to follow above mentioned steps to update records with “Pending” to “Error” status.

Note: These is an online process for bulk update, but it could have a performance impact.

7.2 File Request Status Update Batch

This batch can be used for bulk record status update from “Error” to “Retry” status or “Pending” to “Error” status.

8. Transforming Data

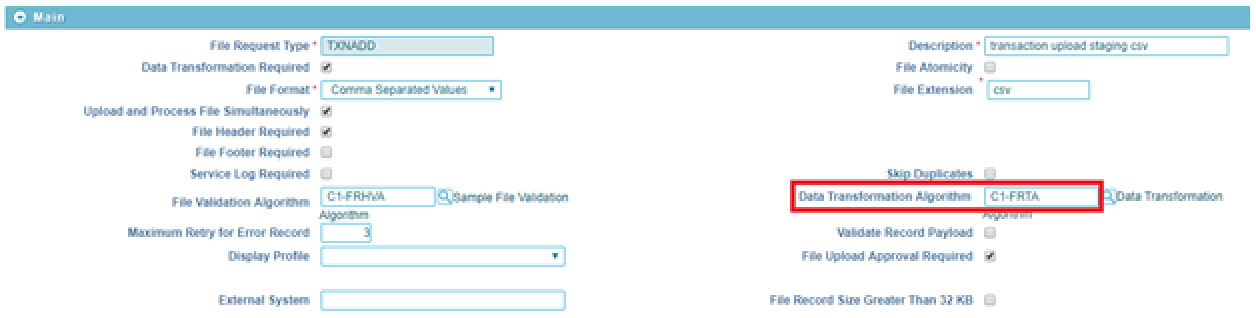
8.1 Transforming File Record in Client Supported Format into ORMB Conform XML

You can transform files in Comma Separated Values (CSV) or Extensible Markup Language (XML) or Fixed Position or JavaScript Object Notation (JSON) or Pipe Separated Values (PSV) or Tilde Separated Values (TSV) formats to ORMB conforming XML using "Data Transformation Algorithm".

"Data Transformation Algorithm" implements "FileRequestTranformationAlgorithmSpot".

To transform file record, specify File Request Transformation algorithm in Data Transformation Algorithm field present in Main section. You can also use the Search ( ) icon corresponding to Data Transformation Algorithm field.

Note: Ensure that Data Transformation Required field is selected before specifying the algorithm.


Main

File Request Type *	TXNADD	Description *	transaction upload staging csv
Data Transformation Required	✓	File Atomicity	☐
File Format *	Comma Separated Values	File Extension *	csv
Upload and Process File Simultaneously	✓		
File Header Required	✓		
File Footer Required	☐		
Service Log Required	☐	Skip Duplicates	☐
File Validation Algorithm	C1-FRHVA Sample File Validation Algorithm	Data Transformation Algorithm	C1-FRTA Data Transformation Algorithm
Maximum Retry for Error Record	3	Validate Record Payload	☐
Display Profile		File Upload Approval Required	✓
External System		File Record Size Greater Than 32 KB	☐

Figure 50: Data Transformation Algorithm

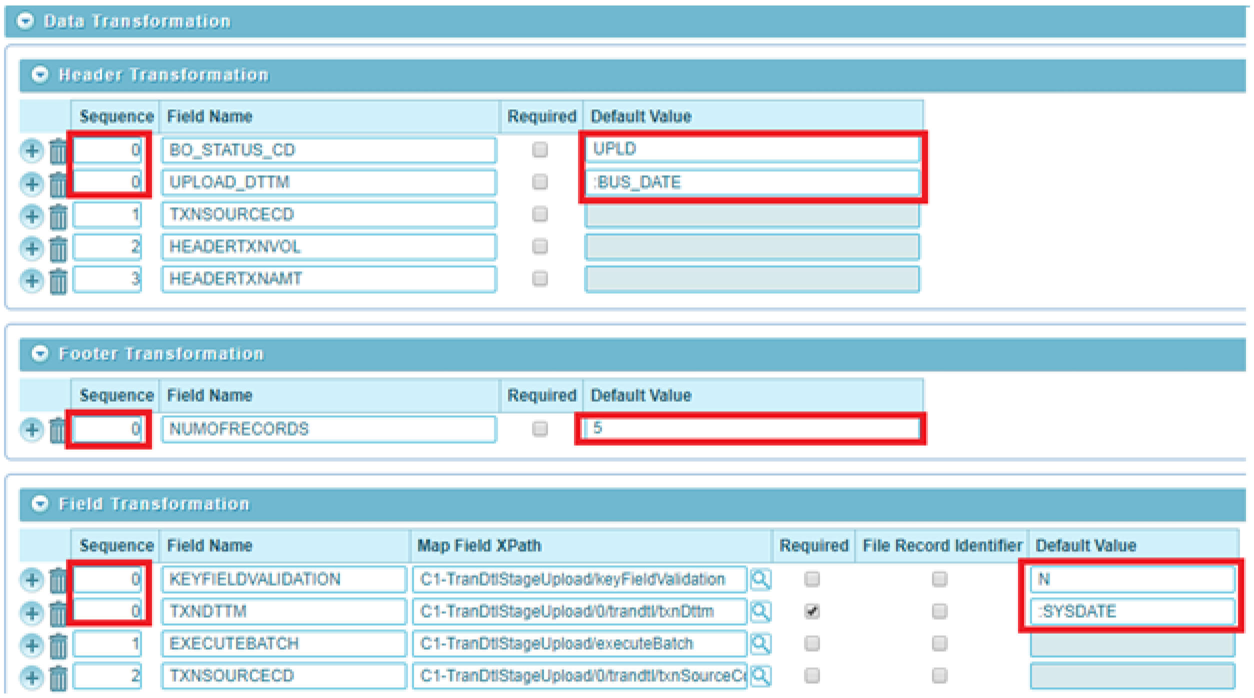
Product provides default transformation algorithm capabilities of the same are listed below.

8.2 Configuring a Default Value in Data Transformation

You can configure default value corresponding to a field. The default value for Sequence should be zero.

Note: Sequence zero is reserved for default value configuration.

To set default value for fields with 'Date' as datatype, you can either provide a date value in following date format 'yyyy-MM-dd / yyyy-MM-dd-HH.mm.ss'or the predefined values listed below.



Data Transformation

Header Transformation

Sequence	Field Name	Required	Default Value
0	BO_STATUS_CD	☐	UPLD
0	UPLOAD_DTTM	☐	:BUS_DATE
1	TXNSOURCECD	☐	
2	HEADERTXNVOL	☐	
3	HEADERTXNAMT	☐	

Footer Transformation

Sequence	Field Name	Required	Default Value
0	NUMOFRECORDS	☐	5

Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation	☐	☐	N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓	☐	:SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch	☐	☐	
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC	☐	☐	

Figure 51: Data Transformation - Default Value

- :BUS_DATE for Business Date - relates to process date
- :SYSDATE for System date - relates to System Date
- :STD_DATE for Standard date - relates to LOCALE date
- :BUS_DTTM for Business Date time - relates to process date time
- :SYS_DTTM for System date time - relates to System Date Time
- :STD_DTTM for Standard date time - relates to LOCALE date time

To update File Request Type detail, click Edit ( ) icon. This enables the 'Default Value' field and fetches the datatype of that corresponding field. The updated value is then used to set the default value.

8.2.1 Applying Default Values Set in File Validation Algorithm to a Field

File Upload Interface provides you with set of five parameters which you can use while transforming records. These parameters are defined using setter methods for default1/2/3/4/5 in file validation algorithm.

Prerequisite

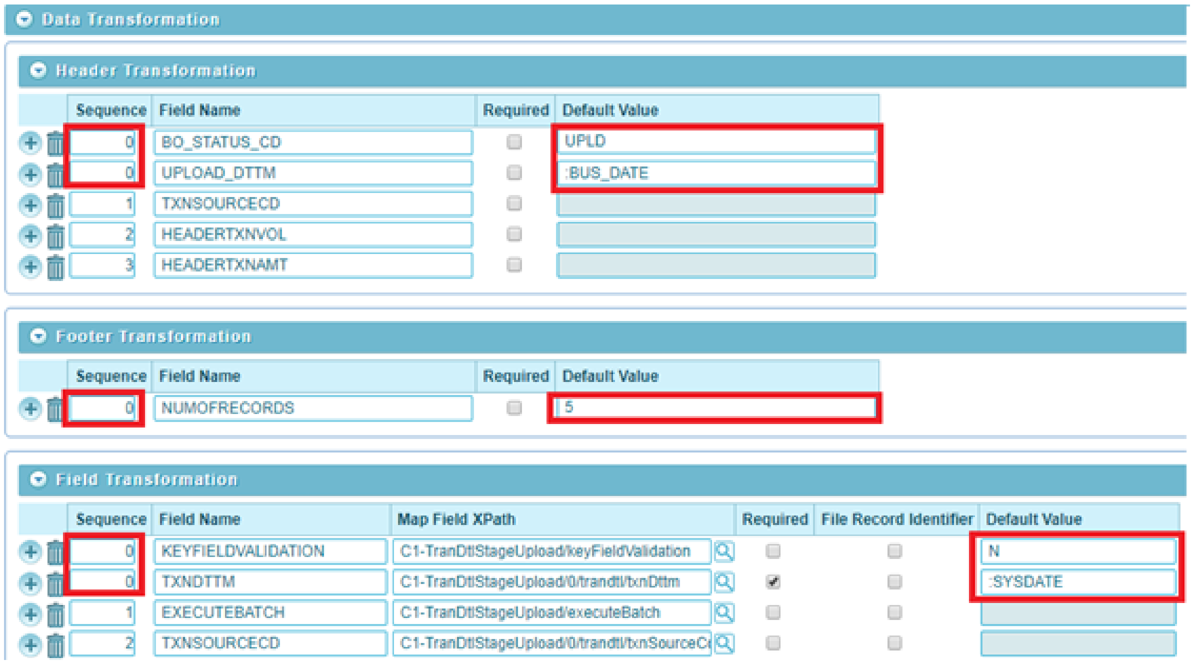
To refer default parameter values, you should have a File Validation Algorithm attached in Main section.

Procedure

To refer these default parameters, you can use either of the following constants in Default Value field within Transformation Details section.

- :DEFAULT1/2/3/4/5
- :SYSDATE/SYS_DTTM
- :BUS_DATE/BUS_DTTM
- :STD_DATE/STD_DTTM

When you use any of the above constants, the value corresponding to the defined parameter is set as default value to the respective field name.



Sequence	Field Name	Required	Default Value
0	BO_STATUS_CD	☐	UPLD
0	UPLOAD_DTTM	☐	:BUS_DATE
1	TXNSOURCECD	☐	
2	HEADERTXNVOL	☐	
3	HEADERTXNAMT	☐	

Sequence	Field Name	Required	Default Value
0	NUMOFRECORDS	☐	5

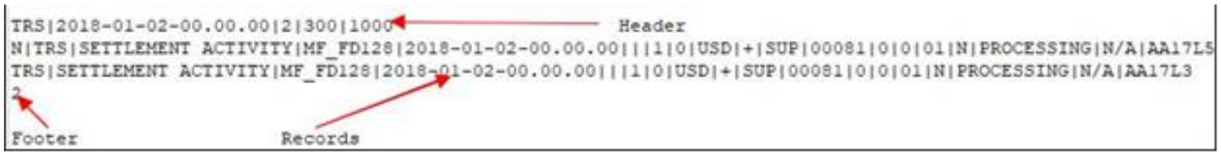
Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation	☐	☐	N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓	☐	:SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch	☐	☐	
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC	☐	☐	

Figure 52: Applying Default Values Set in File Validation Algorithm to a Field

8.3 Mapping Data Line or Record in a File

You can map record in a file using Data Transformation section. Using Data Transformation configuration, mapping can be done only for files in CSV or XML or Fixed Position or JavaScript Object Notation (JSON) or Pipe Separated Values (PSV) or Tilde Separated Values (TSV) formats.

A sample file in PSV format is represented below. The entries are separated by '|' (pipe).



```
TRS|2018-01-02-00.00.00|2|300|1000
N|TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00|||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L5
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00|||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L3
2
```
Header
Footer
Records

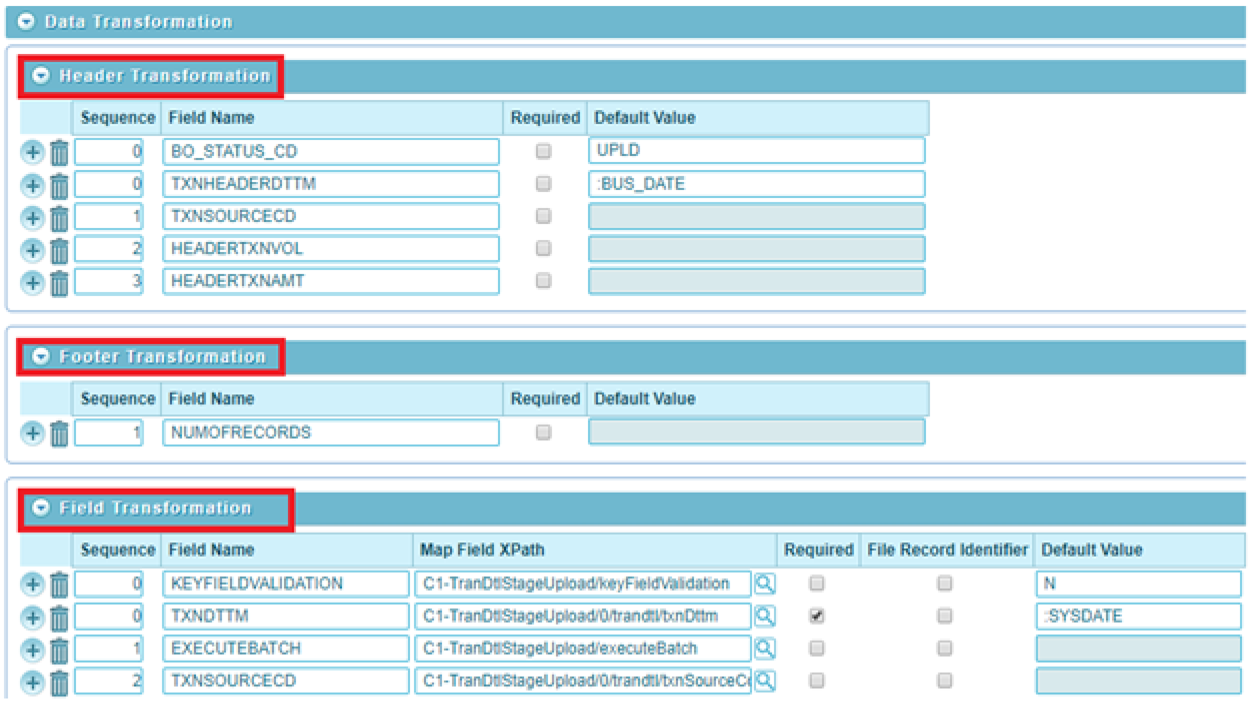
Figure 53: Sample_1.dat

File is divided into three segments:

- File Header - First row in the file
- File record(s)
- File Footer - Last row in the file

The File Request Type-Data Transformation is divided in 3 sections:

- Header Transformation
- Footer Transformation
- Field Transformation



Data Transformation

Header Transformation

Sequence	Field Name	Required	Default Value
0	BO_STATUS_CD		UPLD
0	TXNHEADERDTTM		:BUS_DATE
1	TXNSOURCECD		
2	HEADERTXNVOL		
3	HEADERTXNAMT		

Footer Transformation

Sequence	Field Name	Required	Default Value
1	NUMOFRECORDS		

Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation			N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓		:SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch			
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC			

Figure 54: Mapping data line

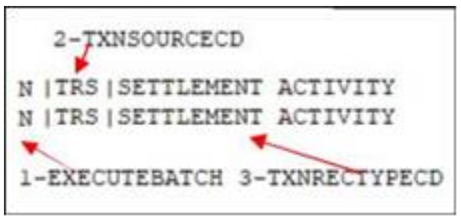
All three segments in data transformation configuration are independent of each other. You can configure default values for any of these segments.

Multiple number of file records should have the same number of fields. For example, if a file has 10 records with seven fields, then all those 10 records should have seven fields, with or without values.

There are two different ways to configure data line mapping:

- For files in CSV or PSV or TSV format:
  - Mapping is done by defining **Sequence**

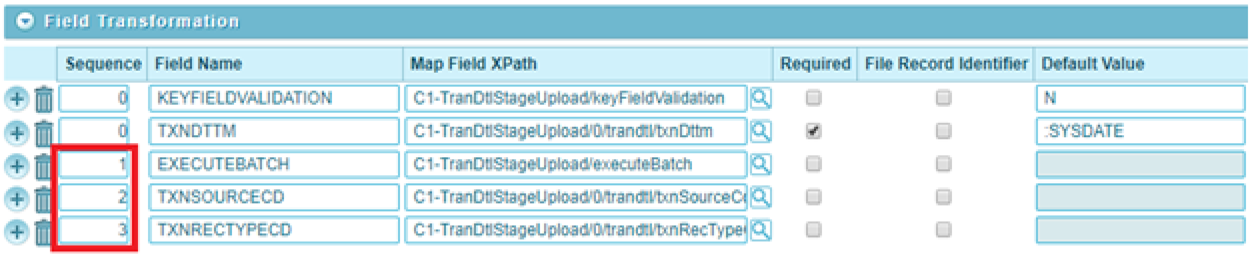
A sample file is represented below:



```
  2-TXNSOURCECD
N |TRS |SETTLEMENT ACTIVITY
N |TRS |SETTLEMENT ACTIVITY

1-EXECUTEBATCH 3-TXNRECTYPECD
```

Figure 55: Sample File.txt

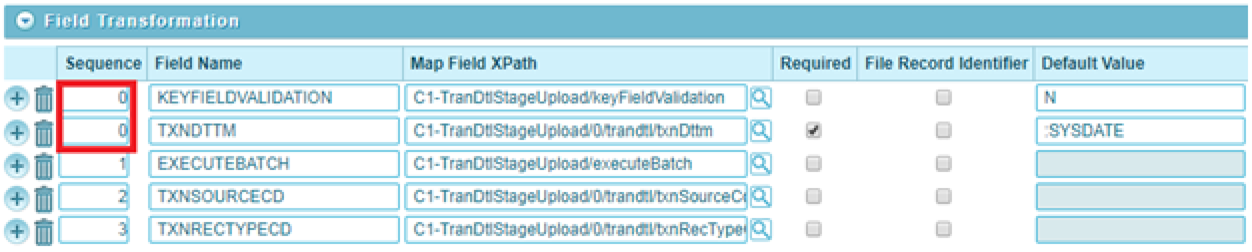


Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation			N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓		:SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch			
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC			
3	TXNRECTYPECD	C1-TranDtlStageUpload/0/trandtl/txnRecType			

Figure 56: Sample File Mapping

- For file data mapping, Sequence index starts with 1
- Sequence with Zero index is reserved for mapping default value



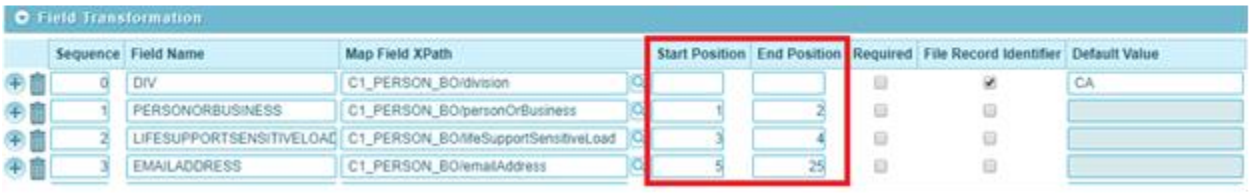
Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation			N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓		:SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch			
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC			
3	TXNRECTYPECD	C1-TranDtlStageUpload/0/trandtl/txnRecType			

Figure 57: Sample File Mapping Default Value

- Same Sequence mapped for multiple fields

- For files in 'Fixed Position' format:
  - Mapping is done using 'Start Position' and 'End Position'.



Field Transformation

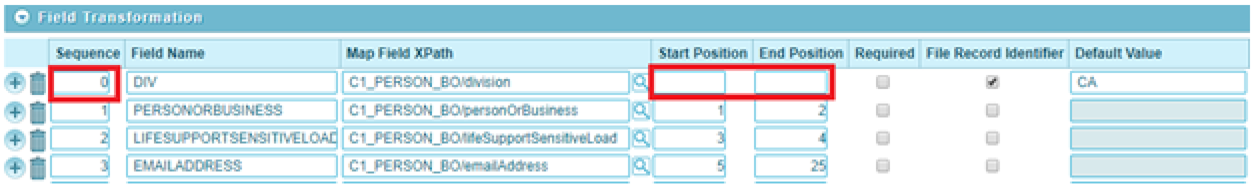
Sequence	Field Name	Map Field XPath	Start Position	End Position	Required	File Record Identifier	Default Value
0	DIV	C1_PERSON_BO/division				✓	CA
1	PERSONORBUSINESS	C1_PERSON_BO/personOrBusiness	1	2			
2	LIFESUPPORTSENSITIVELOAD	C1_PERSON_BO/lifeSupportSensitiveLoad	3	4			
3	EMAILADDRESS	C1_PERSON_BO/emailAddress	5	25			

Figure 58: File Mapping – Fixed Position

- Start Position index starts with 1

Note that it is not required to define 'Start Position' and 'End Position'.

- Sequence with **Zero** index is used for mapping default value



Field Transformation

Sequence	Field Name	Map Field XPath	Start Position	End Position	Required	File Record Identifier	Default Value
0	DIV	C1_PERSON_BO/division				✓	CA
1	PERSONORBUSINESS	C1_PERSON_BO/personOrBusiness	1	2			
2	LIFESUPPORTSENSITIVELOAD	C1_PERSON_BO/lifeSupportSensitiveLoad	3	4			
3	EMAILADDRESS	C1_PERSON_BO/emailAddress	5	25			

Figure 59: File Mapping – Sequence Index

- Same positions can be mapped for multiple fields

8.3.1 Mapping XML/JSON Data Line or Record in a File

You can map record in a XML file using Data Transformation section. A sample file in XML format is represented below.

```
<root>
    <header>
        <personName>MaheshK</personName>
        <accountNo>4321</accountNo>
    </header>
    <footer>
        <numberOfRecords>4</numberOfRecords>
    </footer>
    <request>
        <Person>
            <personOrBusinessClient>R</personOrBusinessClient>
            <divisionClient>CA</divisionClient>
            <personNameClient>
                <nameTypeClient>PRIM</nameTypeClient>
            </personNameClient>
        </Person>
    </request>
</root>
```

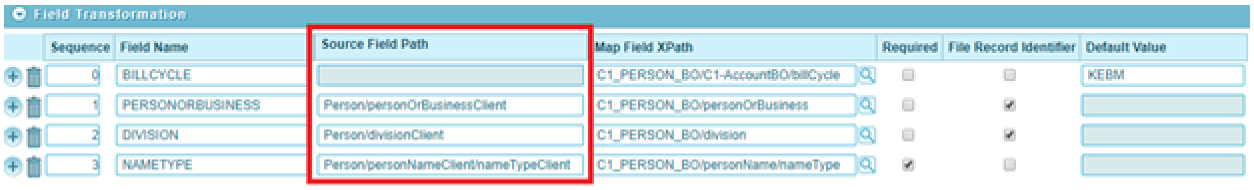
Figure 60: Sample_1.dat

Every record in XML file is divided into three blocks:

- **Header**: Defined in a block having tag name that is defined as **Header XML Tag** in its corresponding File Request Type
- **Service Data**: Defined in a block having tag with service/schema name
- **Footer**: Defined in a block having tag name that is defined as **Footer XML Tag** in its corresponding File Request Type

For XML/JSON record mapping, Source Field Path in Field Transformation is used to hold field xpath in legacy XML/JSON file.

- Source Field Path in Field Transformation is visible only if file format is either **XML** or **JSON**

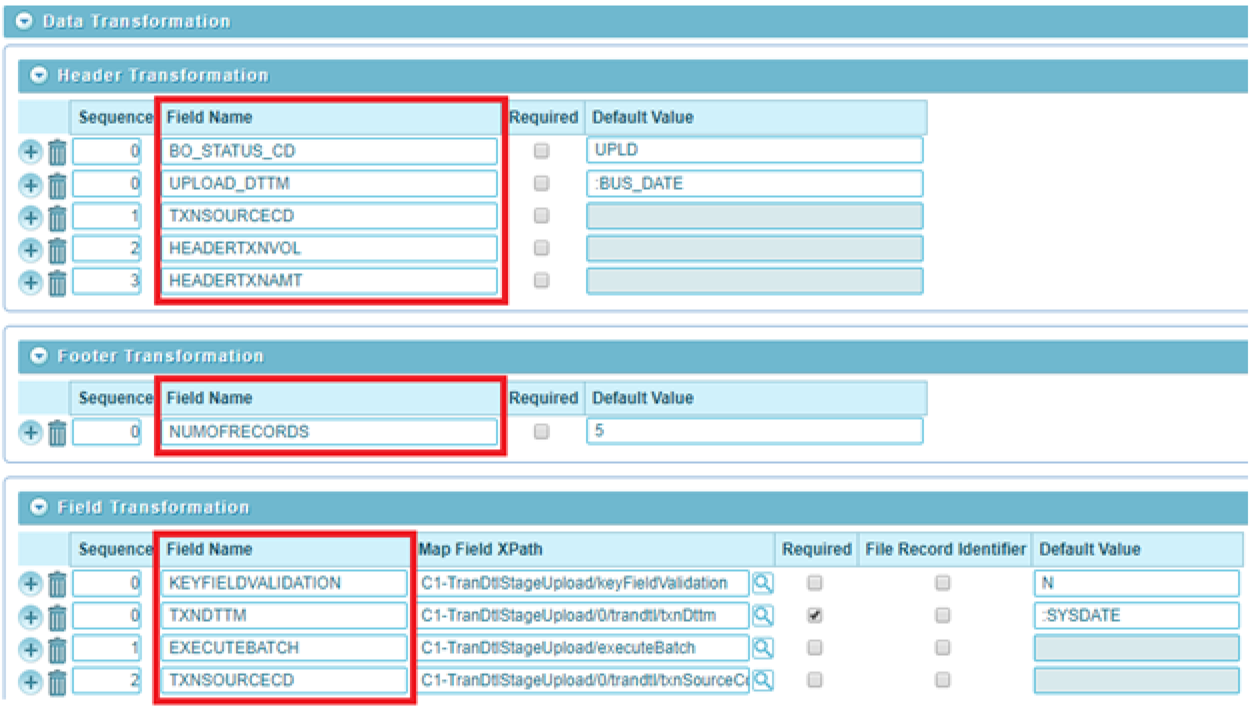


Sequence	Field Name	Source Field Path	Map Field XPath	Required	File Record Identifier	Default Value
0	BILLCYCLE		C1_PERSON_BO/C1-AccountBO/billCycle	☐	☐	KEBM
1	PERSONORBUSINESS	Person/personOrBusinessClient	C1_PERSON_BO/personOrBusiness	☐	✓	
2	DIVISION	Person/divisionClient	C1_PERSON_BO/division	☐	✓	
3	NAMETYPE	Person/personNameClient/nameTypeClient	C1_PERSON_BO/personName/nameType	✓	☐	

Figure 61: Field Transformation

8.4 Mapping File Sequence

This is a unique user defined field, which holds the mapped file sequence field value. It is used as column identifier of a file record. It is free text and can have any user defined value. It has to be uniquely defined within its respective sections i.e. Header Transformation or Footer Transformation or Field Transformation.



Header Transformation

Sequence	Field Name	Required	Default Value
0	BO_STATUS_CD	☐	UPLD
0	UPLOAD_DTTM	☐	:BUS_DATE
1	TXNSOURCECD	☐	
2	HEADERTXNVOL	☐	
3	HEADERTXNAMT	☐	

Footer Transformation

Sequence	Field Name	Required	Default Value
0	NUMOFRECORDS	☐	5

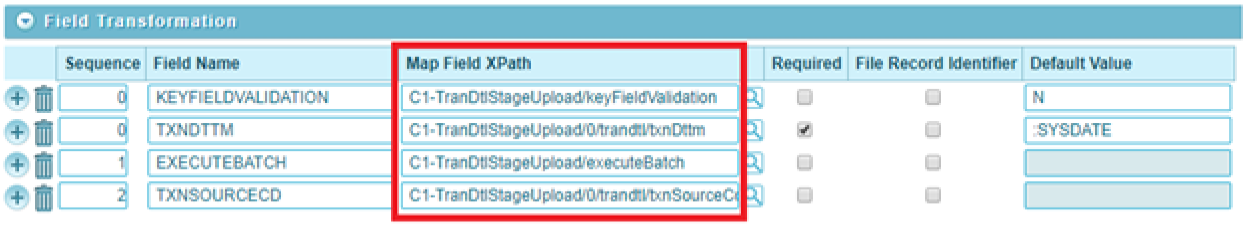
Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation	☐	☐	N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓	☐	:SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch	☐	☐	
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC	☐	☐	

Figure 62: Data Transformation – Field Name

8.5 Using Map Field XPath to Transform Records

The Map Field XPath uses ORMB application provided sample Data Transformation Algorithm to transform file record into ORMB conform XML. If you want to use this algorithm, you must provide "Map Field XPath" for every configured field.

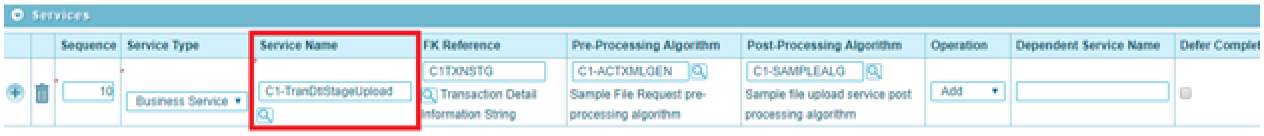

Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation	☐	☐	N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓	☐	:SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch	☐	☐	
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC	☐	☐	

Figure 63: Data Transformation - Map Field Xpath

Note: The Header Transformation and Footer Transformation does not have 'Map Field XPath' column.

These will have an 'XPath' value with reference to its configured service schema.


Services

Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Post-Processing Algorithm	Operation	Dependent Service Name	Defer Complet
10	Business Service	C1-TranDtlStageUpload	C1TXNSTG Transaction Detail Information String	C1-ACTXMLGEN Sample File Request pre-processing algorithm	C1-SAMPLEALG Sample file upload service post processing algorithm	Add		☐

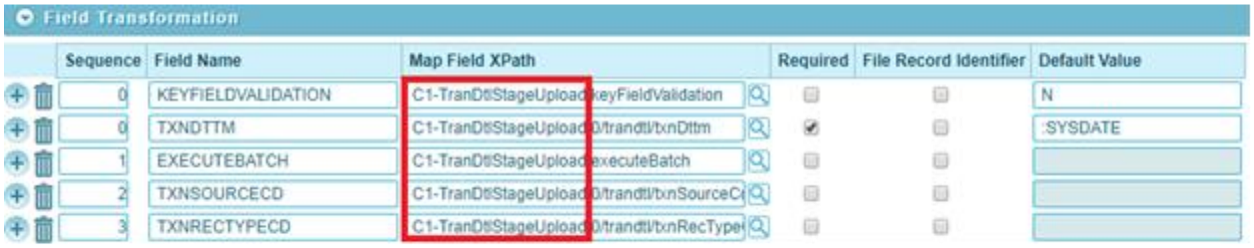
Figure 64: Service Name – Xpath Value

Note: You can search 'xpath' values using the Search ( ) icon or enter a free text.

Index should be used to configure child entity element xpath in a service schema. To configure child entity element xpath, you need to follow below steps:

1. Click Search corresponding to Map Field XPath. File request transform map field zone appears.
2. Select Service Type from the drop-down list.
3. Enter Service Name.

Note: You can select only the child element xpath with index value zero. If you require more than one child elements for a single entity, then you need to select child element xpath with zero' index and edit the required index. For example, '1','2', etc.



Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation			N
0	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓		SYSDATE
1	EXECUTEBATCH	C1-TranDtlStageUpload/executeBatch			
2	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC			
3	TXNRECTYPECD	C1-TranDtlStageUpload/0/trandtl/txnRecType			

Figure 65: Map Field Xpath

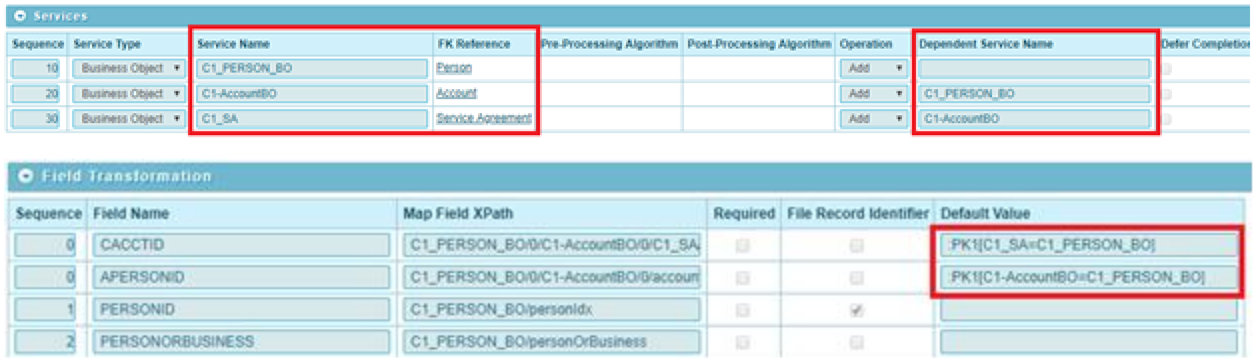
8.6 Link Parent Service Output with Child Service Input

File Upload Interface provides you the flexibility to link parent service output (Primary) key values to child service input field values while transforming data. The supported mapping format is **':PK1/2/3/4/5[CHILD_SERVICE_NAME=PARENT_SERVICE_NAME]'**

For example, there are three business objects which form a hierarchy:

- Person
- Account
- Service Agreement

To map 'Per_Id' within 'Account' business object with primary key value of 'Person' business object (Per_Id), enter the value **:PK1[C1-AccountBO=C1_PERSON_BO]** in **Default Value** field corresponding to Person ID field.

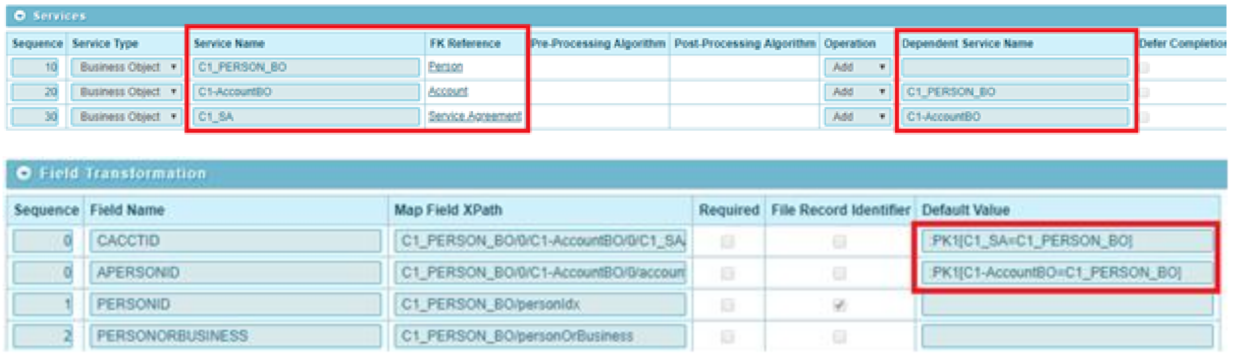


Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Post-Processing Algorithm	Operation	Dependent Service Name	Defer Completion
10	Business Object	C1_PERSON_BO	Person			Add		
20	Business Object	C1-AccountBO	Account			Add	C1_PERSON_BO	
30	Business Object	C1_SA	Service Agreement			Add	C1-AccountBO	

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	CACCTID	C1_PERSON_BO/0/C1-AccountBO/0/C1_SA			:PK1[C1_SA=C1_PERSON_BO]
0	APERSONID	C1_PERSON_BO/0/C1-AccountBO/0/accoun			:PK1[C1-AccountBO=C1_PERSON_BO]
1	PERSONID	C1_PERSON_BO/personIdx		✓	
2	PERSONORBUSINESS	C1_PERSON_BO/personOrBusiness			

Figure 66: Mapping Primary Key Value of 'Person' within 'Account' Business Object

To map 'C1_SA' within 'Service Agreement' business object with primary key value of 'Account' business object, enter the value **:PK1[C1_SA=C1-AccountBO]** in Default Value field corresponding to Account ID field.



Services

Sequence	Service Type	Service Name	FK Reference	Pre-Processing Algorithm	Post-Processing Algorithm	Operation	Dependent Service Name	Defer Completio
10	Business Object	C1_PERSON_BO	Person			Add		☐
20	Business Object	C1-AccountBO	Account			Add	C1_PERSON_BO	☐
30	Business Object	C1_SA	Service Agreement			Add	C1-AccountBO	☐

Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	CACCTID	C1_PERSON_BO/0/C1-AccountBO/0/C1_SA	☐	☐	:PK1[C1_SA=C1_PERSON_BO]
0	APERSONID	C1_PERSON_BO/0/C1-AccountBO/0/accoun	☐	☐	:PK1[C1-AccountBO=C1_PERSON_BO]
1	PERSONID	C1_PERSON_BO/personIdx	☐	✓	
2	PERSONORBUSINESS	C1_PERSON_BO/personOrBusiness	☐	☐	

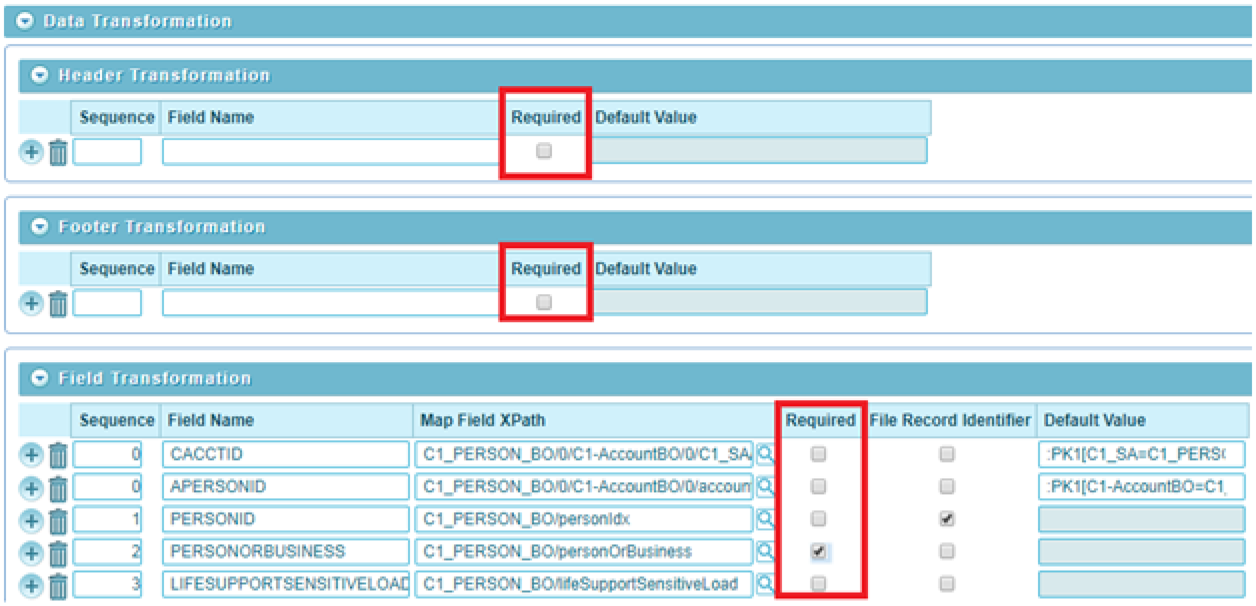
Figure 67: Mapping Primary Key Value of 'Service Agreement' within 'Account' Business Object

Linking parent service output with child service input helps to avoid use of pre-processing algorithm.

8.7 Validating Input Values

You can perform mandatory field level validations for every record while uploading data using File Transform and Upload (C1-FTRAN) Batch.

To perform field level mandatory validations, select Required check box for those required fields in its corresponding section.



Data Transformation

Header Transformation

Sequence	Field Name	Required	Default Value
		☐	

Footer Transformation

Sequence	Field Name	Required	Default Value
		☐	

Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Default Value
0	CACCTID	C1_PERSON_BO/0/C1-AccountBO/0/C1_SA	☐	☐	:PK1[C1_SA=C1_PERS
0	APERSONID	C1_PERSON_BO/0/C1-AccountBO/0/accoun	☐	☐	:PK1[C1-AccountBO=C1
1	PERSONID	C1_PERSON_BO/personIdx	☐	✓	
2	PERSONORBUSINESS	C1_PERSON_BO/personOrBusiness	✓	☐	
3	LIFESUPPORTSENSITIVELOAD	C1_PERSON_BO/lifeSupportSensitiveLoad	☐	☐	

Figure 68: Validating Input Values

9. File Management System

9.1 Overview of files uploaded on SFTP server

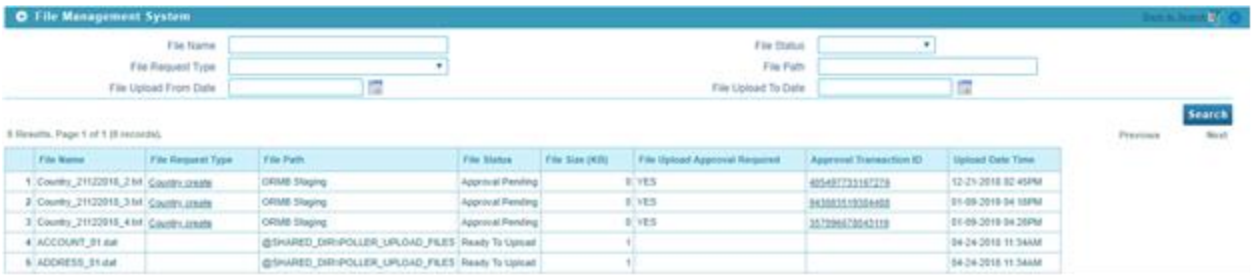
You can have an overview of list of files uploaded on SFTP server. In addition to these, this list will also have those files in ORMB staging with 'Pending' and 'Approval Pending' status i.e. files that has been uploaded in ORMB staging but not processed.

File overview is categorized into four different file status,

- **Ready To Upload**: These files available to upload in ORMB staging
- **Copy in Progress**: File upload on SFTP server is in progress
- **Approval Pending**: File uploaded in ORMB staging and awaiting for approval required to process the file
- **Pending to Process**: File uploaded in ORMB staging and now available to process

List of files to be displayed with following details,

- **File Name** – Name of the file
- **File Request Type** – This column have value for only those files that has already been uploaded in ORMB staging with 'Pending' and 'Approval Pending' status
- **File Path** – File Path for files that have already been uploaded in ORMB staging is always 'ORMB Staging'
- **File Status** - Refer the list above
- **File Size** – File Size is shown for only those files that has not yet uploaded in ORMB staging
- **File Upload Approval Required**: Flag to show whether approval is required to process this file
- **Approval Transaction ID**: Approval transaction ID for the file (You can navigate to the corresponding approval transaction)
- **Upload Date Time**: File upload date time on SFTP server



5 Results. Page 1 of 1 (5 records).

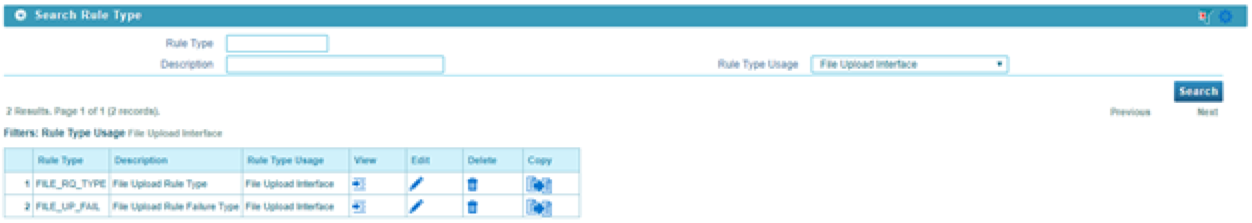
	File Name	File Request Type	File Path	File Status	File Size (KB)	File Upload Approval Required	Approval Transaction ID	Upload Date Time
1	Country_21122018_2.txt	Country create	ORMB Staging	Approval Pending	0	YES	405497731167278	12-21-2018 02:45PM
2	Country_21122018_3.txt	Country create	ORMB Staging	Approval Pending	0	YES	943083519358468	01-09-2019 04:18PM
3	Country_21122018_4.txt	Country create	ORMB Staging	Approval Pending	0	YES	357096678043119	01-09-2019 04:26PM
4	ACCOUNT_01.dat		@SHARED_DIR\POLLER_UPLOAD_FILES	Ready To Upload	1			04-24-2018 11:34AM
5	ADDRESS_01.dat		@SHARED_DIR\POLLER_UPLOAD_FILES	Ready To Upload	1			04-24-2018 11:34AM

Figure 69: File Management System

10. Rule configuration for SFTP poller batch

SFTP poller batch has the flexibility to poll the files on SFTP server that are to be uploaded in ORMB staging using rule configuration. There are two predefined rule types used in SFTP poller batch execution:

- FILE_RQ_TYPE - File Upload Rule Type
- FILE_UP_FAIL - File Upload Rule Failure Type



Rule Type Usage: File Upload Interface

2 Results. Page 1 of 1 (2 records).

Filters: Rule Type Usage File Upload Interface

	Rule Type	Description	Rule Type Usage	View	Edit	Delete	Copy
1	FILE_RQ_TYPE	File Upload Rule Type	File Upload Interface				
2	FILE_UP_FAIL	File Upload Rule Failure Type	File Upload Interface				

Figure 70: Rule Type

10.1 Rule Type for polling files on SFTP server

Prerequisites

- DBMS Scheduler Job should be defined for post execution of the required batch program
- Email configuration using Admin Menu > M > Master Configuration > Email Sender Configuration

File Upload Rule Type (FILE_RQ_TYPE) has the rules defined for polling those files that should be uploaded in ORMB staging. Available rule Input and Output parameters for rule type File Upload Rule Type (FILE_RQ_TYPE) are:

Rule Parameter	Type	Description
File Name (FILE_NAME)	Input	This field will have the name of the picked file.
File Path (FILE_PATH)	Input	This field will have the directory path of the picked file.
File Extension (FILE_EXTENSION)	Input	This field will have the extension of the picked file i.e. txt, csv, xml etc.
Directory Name (DIRECTORY_NAME)	Input	This field will have the directory name of the picked file.
Number Of Files (NUMBER_OF_FILES)	Input	This field will have the count for number of files available in the directory of picked file.
Directory Contains File (DIRECTORY_CONTAINS_FILE)	Input	This field will have the comma separated list of file names that are available in the same directory that of the picked file.
No Of Supported Extension Files (NO_OF_SUPPORT_EXTN_FILES)	Input	This field will have the count for number of only those files having the extension that is defined in those configured File Request Type's.

Rule Parameter	Type	Description
No Of Upload Confirmation Files (NO_OF_UPLD_CONFIRM_FILES)	Input	This field will have the count for number of only those files having the extension that is not defined in those configured File Request Type's.
Alert Notification Time (ALERT_NOTIFY_TIME)	Input	This field will have the current execution time (Format-HH:mm).
Job Name (JOB_NAME)	Output	This field should have the Job name that has the batch program that is required to be executed.
Dependent Job Name (DEPENDENT_JOB_NAME)	Output	This field should have the dependent Job Name i.e. the job name that has been already done which will allow to execute the current job defined against Job Name (JOB_NAME)
Dependent Job Interval In Minutes (DEPENDENT_JOB_INTERVAL_MIN)	Output	This field should have the execution time interval in minutes of dependent Job i.e. Dependent job done within this period of time.
Message Category (MESSAGE_CAT_NBR)	Output	This field should have message category number required for creation of email body content.
Message Number (MESSAGE_NBR)	Output	This field should have message number that has the email body content.
Parameter 1 (MESSAGE_PARM1)	Output	This field should have the message parameter value 1.
Parameter 2 (MESSAGE_PARM2)	Output	This field should have the message parameter value 2.
Parameter 3 (MESSAGE_PARM3)	Output	This field should have the message parameter value 3.
Parameter 4 (MESSAGE_PARM4)	Output	This field should have the message parameter value 4.
Parameter 5 (MESSAGE_PARM5)	Output	This field should have the message parameter value 5.
Notify Email (NOTIFY_EMAIL)	Output	This field should have value either 'true' or 'false' which will decide to send an email notification
Email To Recipients (EMAIL_TO_RECIPIENTS)	Output	This field should have comma separated list of email 'To' recipients.

Email Cc Recipients (EMAIL_CC_RECIPIENTS)	Output	This field should have comma separated list of email 'CC' recipients.
Subject (EMAIL_SUBJECT)	Output	This field should have the subject line of the email to be send.
Email Body Content (EMAIL_BODY_CONTENT)	Output	This field should have the email body content of the email to be send.
Last Notified Time (LAST_NOTIFIED_TIME)	Output	This field value will allow us to avoid resending of multiple mails or creation of TODO on succession of the same rule. This field should have the time (Format-HH:mm) that is in sync with that of alert notification time provided in rule condition i.e. Alert Notification Time (ALERT_NOTIFY_TIME) after a reach (>=) of this time mail will be triggered.
To Do Type (TODO_TYPE)	Output	This field should have the To Do Type for which a To Do will be created.
Role ID (TODO_ROLE)	Output	This field should have the To Do Role used for creating To Do.
Drill Key 1 (DRILL_KEY1)	Output	This field should have the drill key 1 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 2 (DRILL_KEY2)	Output	This field should have the drill key 2 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 3 (DRILL_KEY3)	Output	This field should have the drill key 3 corresponding to the given To Do Type which will be used for creating To Do.
Sort Key (SORT_KEY)	Output	This field should have the sort key corresponding to the given To Do Type which will be used for creating To Do.
Assign to User (ASSIGNED_TO)	Output	This field should have the User Id to which a created To Do will be assigned

Note: All those Rule output parameters corresponding to Message Category and Number are optional. You can use them as per the requirement to form an email body content.

BATCH_NUM: This is the constant having current batch number as a value. This constant can be used as a Rule output parameter value for any of the 'DRILL_KEY1/2/3' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

FILE_NAME: This is the constant having input file name as a value. This constant can be used as a Rule output parameter value for any of the 'MESSAGE_PARM1/2/3/4/5' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

10.2 Rule Type for failure notification

Prerequisite

- Email configuration using Admin Menu>>M>>Master Configuration>>Email Sender Configuration

File Upload Rule Failure Type (FILE_UP_FAIL) rule type has the rules defined for sending email notification for the failure in file polling using File Upload Rule Type (FILE_RQ_TYPE). Available rule Input and Output parameters for rule type File Upload Rule Failure Type (FILE_UP_FAIL) are:

Rule Parameter	Type	Description
File Name (FILE_NAME)	Input	This field will have the name of the picked file.
File Path (FILE_PATH)	Input	This field will have the directory path of the picked file.
File Extension (FILE_EXTENSION)	Input	This field will have the extension of the picked file i.e. txt, csv, xml etc.
Directory Name (DIRECTORY_NAME)	Input	This field will have the directory name of the picked file.
Number Of Files (NUMBER_OF_FILES)	Input	This field will have the count for number of files available in the directory of picked file.
Directory Contains File (DIRECTORY_CONTAINS_FILE)	Input	This field will have the comma separated list of file names that are available in the same directory that of the picked file.
No Of Supported Extension Files (NO_OF_SUPPORT_EXTN_FILES)	Input	This field will have the count for number of only those files having the extension that is defined in those configured File Request Type's.
No Of Upload Confirmation Files (NO_OF_UPLD_CONFIRM_FILES)	Input	This field will have the count for number of only those files having the extension that is not defined in those configured File Request Types.
Alert Notification Time (ALERT_NOTIFY_TIME)	Input	This field will have the current execution time (Format-HH:mm).

Message Category (MESSAGE_CAT_NBR)	Output	This field should have message category number required for creation of email body content.
Message Number (MESSAGE_NBR)	Output	This field should have message number that has the email body content.

Rule Parameter	Type	Description
Parameter 1 (MESSAGE_PARM1)	Output	This field should have the message parameter value 1.
Parameter 2 (MESSAGE_PARM2)	Output	This field should have the message parameter value 2.
Parameter 3 (MESSAGE_PARM3)	Output	This field should have the message parameter value 3.
Parameter 4 (MESSAGE_PARM4)	Output	This field should have the message parameter value 4.
Parameter 5 (MESSAGE_PARM5)	Output	This field should have the message parameter value 5.
Notify Email (NOTIFY_EMAIL)	Output	This field should have value either 'true' or 'false' which will decide to send an email notification
Email To Recipients (EMAIL_TO_RECIPIENTS)	Output	This field should have comma separated list of email 'To' recipients.
Email Cc Recipients (EMAIL_CC_RECIPIENTS)	Output	This field should have comma separated list of email 'CC' recipients.
Subject (EMAIL_SUBJECT)	Output	This field should have the subject line of the email to be send.
Email Body Content (EMAIL_BODY_CONTENT)	Output	This field should have the email body content of the email to be send.
Last Notified Time (LAST_NOTIFIED_TIME)	Output	This field value will allow us to avoid resending of multiple mails or creation of TODO on succession of the same rule. This field should have the time (Format-HH:mm) that is in sync with that of alert notification time provided in rule condition i.e. Alert Notification Time (ALERT_NOTIFY_TIME) after a reach (>=) of this time mail will be triggered.
Missing File Name (MISSING_FILE_NAME)	Output	This field will have the pipe separated list of expected file names within the current directory of picked file.
Missing File Description (MISSING_FILE_DESC)	Output	This field will have the pipe separated list of file description. This list should be corresponding to that of 'Missing File Name' list. This file description will be used to populate **[%FILE_NAMES]** constant.

Rule Parameter	Type	Description
To Do Type (TODO_TYPE)	Output	This field should have the To Do Type for which a To Do will be created.
Role ID (TODO_ROLE)	Output	This field should have the To Do Role used for creating To Do.
Drill Key 1 (DRILL_KEY1)	Output	This field should have the drill key 1 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 2 (DRILL_KEY2)	Output	This field should have the drill key 2 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 3 (DRILL_KEY3)	Output	This field should have the drill key 3 corresponding to the given To Do Type which will be used for creating To Do.
Sort Key (SORT_KEY)	Output	This field should have the sort key corresponding to the given To Do Type which will be used for creating To Do.
Assign to User (ASSIGNED_TO)	Output	This field should have the User Id to which a created To Do will be assigned

Note: All those Rule output parameters corresponding to Message Category and Number are optional. These can be used as per the requirement to form an email body content.

[%FILE_NAMES] – This is the constant used along with 'MISSING_FILE_DESC' and 'MISSING_FILE_NAME' output fields and will have list of those file description that are missing in the current directory.

This can be used in creating an email content that requires those missing file names to be shared with email recipients.

BATCH_NUM – This is the constant having current batch number as a value. This constant can be used as a Rule output parameter value for any of the 'DRILL_KEY1/2/3' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

FILE_NAME – This is the constant having input file name as a value. This constant can be used as a Rule output parameter value for any of the 'MESSAGE_PARM1/2/3/4/5' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

10.3 Create Rule using Rule Type

1. Go to Admin Menu, select R and then click Rule. A sub-menu appears.
2. Click Add option from the Rule sub-menu. The Rule window appears. This window has following sections:
   - Main
   - Rule Output Parameters
   - Criteria

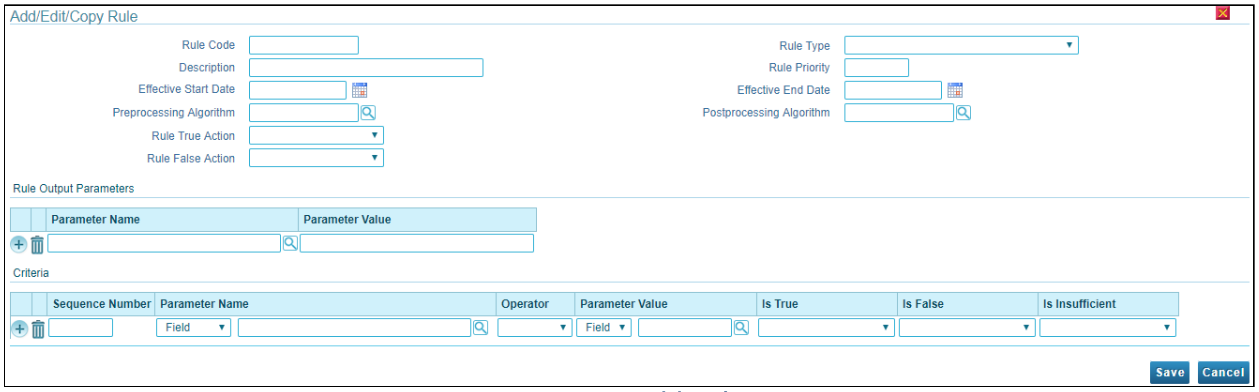
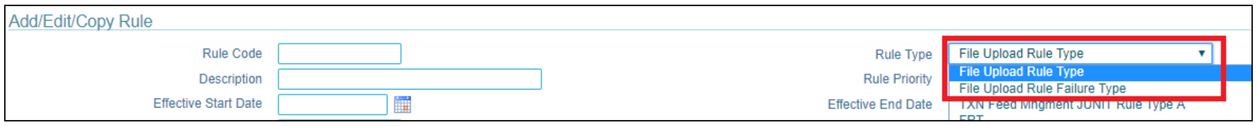


Figure 71: Add Rule

3. Enter unique **Rule Code** in corresponding field.
4. Select either of the Rule Type i.e. **File Upload Rule Failure Type (FILE_UP_FAIL)** or **File Upload Rule Type (FILE_RQ_TYPE)**.



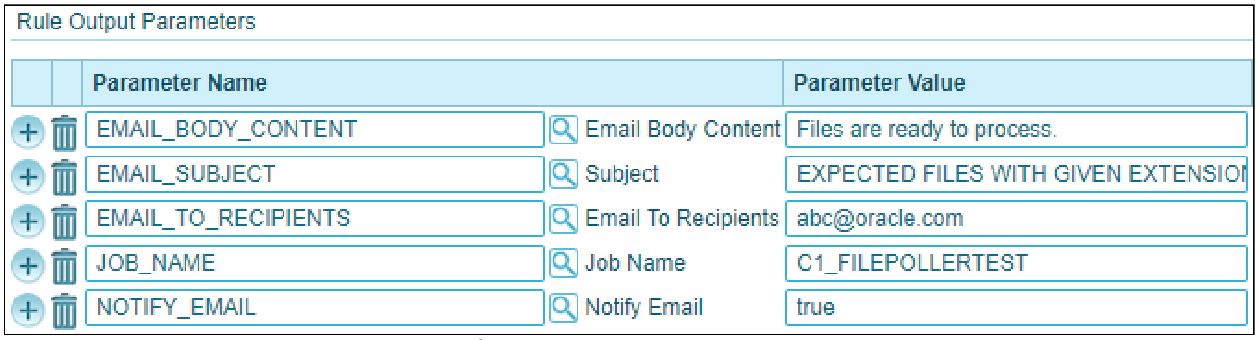
Rule Type: File Upload Rule Type

Figure 72: File Upload Rule Type

5. Enter **Rule Priority**, this will decide the execution **sequence** of this **Rule** for this **Rule Type.**
6. **Effective Start Date** should be less than Business Date and **Effective End Date** should not be blank and greater than **Effective Start Date.**
7. **Rule True Action** should be '**Success'** if we want to conclude with successful verification of this Rule. Else you can select either '**Next Dependent Rule**' or '**Next Rule by Priority**'.
8. **Rule False Action** can be either '**Next Dependent Rule**' or '**Next Rule by Priority**'.

10.3.1 Rule Output Parameters

1. You will have to configure required list of output parameters using the predefined fields of type '**Output**' corresponding to that of selected Rule Type i.e. **File Upload Rule Failure Type (FILE_UP_FAIL)** or **File Upload Rule Type (FILE_RQ_TYPE)**.
2. Rule Output Parameters are the fields with static data values.



Rule Output Parameters

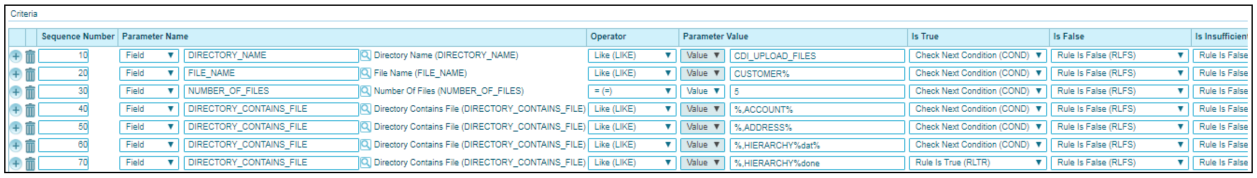
Parameter Name		Parameter Value
EMAIL_BODY_CONTENT	Email Body Content	Files are ready to process.
EMAIL_SUBJECT	Subject	EXPECTED FILES WITH GIVEN EXTENSIO
EMAIL_TO_RECIPIENTS	Email To Recipients	abc@oracle.com
JOB_NAME	Job Name	C1_FILEPOLLERTEST
NOTIFY_EMAIL	Notify Email	true

Figure 73: Rule Output Parameters

3. This list of parameter names and values will be returned if the corresponding **Rule** is successful.

10.3.2 Criteria

1. You can define the criteria required for uploading the picked file into ORMB system.
2. Criteria is the conditions that are formed using '**Input'** type fields defined in corresponding Rule Type; i.e. either **File Upload Rule Failure Type (FILE_UP_FAIL)** or **File Upload Rule Type (FILE_RQ_TYPE)**.



Criteria

Sequence Number	Parameter Name			Operator	Parameter Value		Is True	Is False	Is Insufficient
10	Field	DIRECTORY_NAME	Directory Name (DIRECTORY_NAME)	Like (LIKE)	Value	CDI_UPLOAD_FILES	Check Next Condition (COND)	Rule Is False (RLFS)	Rule Is False
20	Field	FILE_NAME	File Name (FILE_NAME)	Like (LIKE)	Value	CUSTOMER%	Check Next Condition (COND)	Rule Is False (RLFS)	Rule Is False
30	Field	NUMBER_OF_FILES	Number Of Files (NUMBER_OF_FILES)	= (=)	Value	5	Check Next Condition (COND)	Rule Is False (RLFS)	Rule Is False
40	Field	DIRECTORY_CONTAINS_FILE	Directory Contains File (DIRECTORY_CONTAINS_FILE)	Like (LIKE)	Value	%,ACCOUNT%	Check Next Condition (COND)	Rule Is False (RLFS)	Rule Is False
50	Field	DIRECTORY_CONTAINS_FILE	Directory Contains File (DIRECTORY_CONTAINS_FILE)	Like (LIKE)	Value	%,ADDRESS%	Check Next Condition (COND)	Rule Is False (RLFS)	Rule Is False
60	Field	DIRECTORY_CONTAINS_FILE	Directory Contains File (DIRECTORY_CONTAINS_FILE)	Like (LIKE)	Value	%,HIERARCHY%dat%	Check Next Condition (COND)	Rule Is False (RLFS)	Rule Is False
70	Field	DIRECTORY_CONTAINS_FILE	Directory Contains File (DIRECTORY_CONTAINS_FILE)	Like (LIKE)	Value	%,HIERARCHY%done	Rule Is True (RLTR)	Rule Is False (RLFS)	Rule Is False

Figure 74: Criteria

3. Criteria are validated based on the list of input parameters with its corresponding values.
4. Input parameters are populated with in SFTP Poller batch and provided as input while invoking rule engine.

11. Functions used for FRT configuration

FRT has the scope to configure the supported functions that can be used to manipulate the file inputs and enrich the input payload with ORMB identifiers.

List of available functions are:

1. FETCH functions
2. COPY function
3. String Handling
4. SCRIPT function
5. TO DO Generation
6. EVAL function
7. DUPLICATE function
8. Get Feature Configuration value
9. HEADER function
10. Mapping Primary Key value – Refer section Link Parent Service Output with Child Service Input

11.1 FETCH functions

These are generic functions that can be used to get the required entity identifiers or its corresponding attributes that persist in ORMB system.

We can get the different field/column values using single fetch function. Provided FETCH functions that has an input parameter **SET_OPERATION** has got the capability to reset the configured corresponding service level operation.

Function inputs can have the LITERAL values i.e. static values enclosed in single quotes or the CONSTANT values i.e. ":BUS_DATE, :BUS_DTTM , :STD_DATE, :STD_DTTM, :SYSDATE, :SYS_DTTM, :CHAR_DATE, :FILENAME, :FILE_ID" or those configured FIELD names in FIELD TRANSFORMATION whose mapped value will be referred.

Note: To enhance the performance using this fetch function(s) RESULTSET is maintained in that transaction request. So even if the same function is triggered for multiple times with same inputs, we can get the required field value from the RESULTSET in that request transaction.

FETCH functions can be used in only **FIELD TRANSFORMATION** section but not in HEADER and FOOTER TRANSFORMATION.

List of FETCH functions,

- Fetch Account
- Fetch Account Characteristic
- Fetch Account Party UID
- Fetch Adjustment Contract
- Fetch Characteristic Entity Flag
- Fetch Characteristic Type Flag
- Fetch Contract

- Fetch Extendable Lookup value
- Fetch Invoice Construct
- Fetch Lookup
- Fetch Parent Person
- Fetch Price Assignment
- Fetch Person
- Fetch Person Characteristic
- Fetch Price Item Contract
- Fetch Price List
- Fetch Price List Characteristic
- Fetch Price List Assignment ID
- Fetch Person Party UID
- Fetch Person Relationship
- Fetch Rate Map Id
- Fetch User
- Check Access Group

11.1.1 Fetch Account

These function can be used to get and map the account identifier or its corresponding column values in ORMB system to the input payload i.e. manipulate the service payload.

Account identifier can be retrieved for either the given *PER_ID_NBR* i.e. with reference to the defined relationship between Person and Account in "Account Person Relationship" table or *ACCT_NBR* i.e. with reference to the relationship defined in the "Maintains External Account Numbers" table.

Other remaining field inputs are optional. Input fields with no mapping can have "NULL" value in the function.

Function:

```
FETCH(ACCT[COLUMN_NAME,PER_ID_NBR,ID_TYPE_CD,ACCT_REL_TYPE_CD,ACCT_NBR
,ACCT_NBR_TYPE_CD,DIVISION,ACCT_USAGE_FLG,PRIM_SW,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	**Description**	**Table Description**
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
ACCT_REL_TYPE_CD	Account Relationship Type	Account Person Relationship
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
PRIM_SW	Primary ID switch	Person Identifier
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	**Description**	**Table Description**
ACCT_ID	Account ID	Account
CIS_DIVISION	Division of an Account	Account
CURRENCY_CD	Currency Code of an Account	Account
PER_ID	Person ID (Returned only when PER_ID_NBR is passed as an function input)	Person Identifier
ID_TYPE_CD	Person Identifier Type (Returned only when PER_ID_NBR is passed as an function input)	Person Identifier
ACCT_REL_TYPE_CD	Account Relationship Type (Returned only when PER_ID_NBR is passed as an function input)	Account Person Relationship

ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
ACCT_USAGE_FLG	Account Usage	Account
VERSION	Version	Account

11.1.2 Fetch Account Characteristic

These function can be used to get and map the account Identifier or its corresponding Characteristic values in ORMB system to the input payload i.e. manipulate the service payload.

ACCT_NBR input is mandatory and the other remaining field inputs are optional. Input fields with no mapping can have "NULL" value in the function.

Function:

```
FETCH(ACCT_CHAR[COLUMN_NAME,ACCT_NBR,ACCT_NBR_TYPE_CD,CHAR_TYPE_CD,DIV
ISION,ACCT_USAGE_FLG,PRIM_SW,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
CHAR_TYPE_CD	Characteristic Type Code	Account Characteristics
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
PRIM_SW	Primary ID switch	Person Identifier
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
ACCT_ID	Account ID	Account
CIS_DIVISION	Division of an Account	Account
CURRENCY_CD	Currency Code of an Account	Account
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
ACCT_USAGE_FLG	Account Usage	Account
CHAR_TYPE_CD	Characteristic Type Code	Account Characteristics
CHAR_VAL	Characteristic Value	Account Characteristics
EFFDT	Effective Date	Account Characteristics
CHAR_VAL_FK1	Characteristic Value FK1	Account Characteristics
CHAR_VAL_FK2	Characteristic Value FK2	Account Characteristics
CHAR_VAL_FK3	Characteristic Value FK3	Account Characteristics
CHAR_VAL_FK4	Characteristic Value FK4	Account Characteristics
CHAR_VAL_FK5	Characteristic Value FK5	Account Characteristics
VERSION	Version	Account

11.1.3 Fetch Account Party UID

These function can be used to get and map the account identifier, Party UID or its corresponding column values in ORMB system to the input payload i.e. manipulate the service payload.

Account identifier can be retrieved for either the given *PER_ID_NBR* i.e. with reference to the defined relationship between Person and Account in “Account Person Relationship” table or *ACCT_NBR* i.e. with reference to the relationship defined in the “Maintains External Account Numbers” table.

Other remaining field inputs are optional. Input fields with no mapping can have “NULL” value in the function.

Function:

```
FETCH(ACCT_PUID[COLUMN_NAME,PER_ID_NBR,ID_TYPE_CD,ACCT_REL_TYPE_CD,ACC
T_NBR,ACCT_NBR_TYPE_CD,ACCT_ID,DIVISION,ACCT_USAGE_FLG,PRIM_SW,SET_OPE
RATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	

PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
ACCT_REL_TYPE_CD	Account Relationship Type	Account Person Relationship
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
PRIM_SW	Primary ID switch	Person Identifier
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
ACCT_ID	Account ID	Account
CIS_DIVISION	Division of an Account	Account
CURRENCY_CD	Currency Code of an Account	Account
PER_ID	Person ID	Person Identifier
PARTY_UID	Party UID	Party
ACCT_REL_TYPE_CD	Account Relationship Type	Account Person Relationship
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
ACCT_USAGE_FLG	Account Usage	Account
VERSION	Version	Account

11.1.4 Fetch Adjustment Contract

These function can be used to get and map the account identifier, Contract ID or its corresponding column values in ORMB system to the input payload i.e. manipulate the service payload.

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
ADJ_TYP_CD	Adjustment Type Code	Adjustment Type / Adjustment Type Profile
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

Get the Contract ID, Contract Type for the relationship between "SA Type Adjustment Type Profile" and "Adjustment Type / Adjustment Type Profile"

ACCT_NBR is mandatory input and other remaining field inputs are optional. Input fields with no mapping can have "NULL" value in the function.

Function:

```
FETCH(ADJSA[COLUMN_NAME,ACCT_NBR,ACCT_NBR_TYPE_CD,ADJ_TYP_CD,DIVISION,
ACCT_USAGE_FLG,SET_OPERATION])
```

Expected list of input field/column values:

List of output field/column values:

Output Parameter	Description	Table Description
ACCT_ID	Account ID	Account
CIS_DIVISION	Division of an Account	Account
CURRENCY_CD	Currency Code of an Account	Account
SA_ID	Contract ID	SA (Service Agreement)
SA_TYPE_CD	Contract Type	SA (Service Agreement)
ADJ_TYPE_CD	Account Relationship Type	Adjustment Type / Adjustment Type Profile

ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
ACCT_USAGE_FLG	Account Usage	Account
VERSION	Version	SA (Service Agreement)

11.1.5 Fetch Characteristic Entity Flag

These function can be used to get the Characteristic Entity Flag for the given Characteristic Type Code.

CHAR_TYPE is mandatory input for this function.

Function:

```
FETCH(CHAR_ENTITY_FLG[CHAR_TYPE])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
CHAR_TYPE_CD	Characteristic Type Code	Characteristic Type Entity

List of output field/column values:

Output Parameter	Description	Table Description
CHAR_ENTITY_FLG	Characteristic Entity	Characteristic Type Entity

11.1.6 Fetch Characteristic Type Flag

These function can be used to get the Type of CHAR value for the given Characteristic Type Code.

CHAR_TYPE is mandatory input for this function.

Function:

```
FETCH(CHAR_TYPE_FLG[CHAR_TYPE])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
CHAR_TYPE_CD	Characteristic Type Code	Characteristic Type

List of output field/column values:

Output Parameter	Description	Table Description
CHAR_TYPE_FLG	Type of Char Value	Characteristic Type

11.1.7 Fetch Contract

These function can be used to get the Contract ID, Contract Type and other corresponding below listed column values.

Contract identifier can be retrieved using the Contract and Account relationship. Account identifier can be retrieved for either the given *PER_ID_NBR* i.e. with reference to the defined relationship between Person and Account in "Account Person Relationship" table or *ACCT_NBR* i.e. with reference to the relationship defined in the "Maintains External Account Numbers" table.

Either of the inputs *PER_ID_NBR or ACCT_NBR* is mandatory input for this function.

Function:

```
FETCH(SA[COLUMN_NAME,PER_ID_NBR,ID_TYPE_CD,ACCT_REL_TYPE_CD,ACCT_NBR,A
CCT_NBR_TYPE_CD,SA_TYPE_CD,DIVISION,ACCT_USAGE_FLG,PRIM_SW,SET_OPERATI
ON])
```

Expected list of input field/column values:

Input Parameter	**Description**	**Table Description**
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
ACCT_REL_TYPE_CD	Account Relationship Type	Account Person Relationship
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
SA_TYPE_CD	Contract Type	SA (Service Agreement)
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
PRIM_SW	Primary ID switch	Person Identifier
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error	

	if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
ACCT_ID	Account ID	Account
CIS_DIVISION	Division of an Account	Account
CURRENCY_CD	Currency Code of an Account	Account
PER_ID	Person ID (Returned only when PER_ID_NBR is passed as an function input)	Person Identifier
SA_ID	Contract ID	SA (Service Agreement)
ACCT_REL_TYPE_CD	Account Relationship Type (Returned only when PER_ID_NBR is passed as an function input)	Account Person Relationship
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
SA_TYPE_CD	Contract Type	SA (Service Agreement)
ACCT_USAGE_FLG	Account Usage	Account
VERSION	Version	Account

11.1.8 Fetch Extendable Lookup value

These function can be used to get the value from the extendable lookup's CLOB field i.e. XML payload. Last parameter in function i.e. ***VALIDATE*** is a Boolean parameter. If **TRUE**, it will throw an error, if no value is returned for this function. **FALSE** will do nothing.

It has 2 filter criteria to get the required field value.

FILTER 1 – To get the extendable lookup CLOB object i.e. XML payload

Parameters *BUS_OBJ_CD, EXT_LOOKUP_VALUE* and *EXT_LOOKUP_USAGE_FLG* can be used to get the required extendable lookup record.

Parameters *BUS_OBJ_CD* and *EXT_LOOKUP_VALUE* are mandatory inputs and *EXT_LOOKUP_USAGE_FLG* is optional for this function.

FILTER 2 – To get the required element value from this retrieved CLOB object i.e. XML payload

You can provide filter criteria to get the specific element value in the XML payload. Criteria can be defined within **INPUT**{} block. Comma-Separated multiple inputs can be provided.

These can have criteria in following format,

ELEMENT_REF_XPATH=INPUT_VALUE

ELEMENT_REF_XPATH – This will be the element XPATH in the retrieved XML payload of extendable lookup for which you want to filter the XML block data.

INPUT_VALUE - This will be the element on which you want to filter the XML block data. This can be a LITERAL value or you can map the configured FIELD name in FIELD TRANSFORMATION.

Function:

```
FETCH(EXT_LOOKUP[BUS_OBJ_CD,EXT_LOOKUP_VALUE,EXT_LOOKUP_USAGE_FLG,INPU
T{LOOKUP_XPATH1=?;LOOKUP_XPATH2=?},OUTPUT_XPATH,VALIDATE])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
BUS_OBJ_CD	Business Object Code	
EXT_LOOKUP_VALUE	Extendable Lookup Value	
EXT_LOOKUP_USAGE_FLG	Extendable Lookup Usage flag	

List of output field/column values:

OUTPUT_XPATH – This parameter will be the XPATH of the ELEMENT in XML payload of extendable lookup CLOB object. Value of this element will be returned as a result.

11.1.9 Fetch Invoice Construct

These function can be used to get the Invoice Construct Identifier and other corresponding below listed column/output values.

Invoice Contract identifier can be retrieved using Account Identifier i.e. ACCT_NBR. Using this identifier we look for BILLING ACCOUNT ID in Construct table.

ACCT_NBR is mandatory input while the other remaining input parameters are optional for this function.

Function:

```
FETCH(INV_CONSTRUCT[COLUMN_NAME,ACCT_NBR,ACCT_NBR_TYPE_CD,ACCT_ID,TEMP
LATE_PURPOSE_FLG,EFFT_START_DT,EFFT_END_DT,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers

ACCT_ID	Account ID	Maintains External Account Numbers
TEMPLATE_PURPOSE_FLG	Template Purpose Flag	Construct Template
EFFT_START_DT	Effective Start Date	Construct
EFFT_END_DT	Effective End Date	Construct
STATUS_FLG	Status Flag. By default its value will be 'ACTV'.	Construct
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
CONSTRUCT_ID	Construct ID	Construct
TEMPLATE_PURPOSE_FLG	Template Purpose Flag	Construct Template
EFFT_START_DT	Effective Start Date	Construct
EFFT_END_DT	Effective End Date	Construct
CONSTRUCT_DESC	Contract Description	Construct
TYPE_FLG	Type Flag	Construct
VERSION_NUM	Version Number	Construct
BILLING_ACCT_ID	Contract Type	Construct Template
STATUS_FLG	Status Flag	Construct

11.1.10 Fetch Lookup

These function can be used to get the lookup values.

FIELD_NAME is mandatory input while the other remaining input parameters are optional for this function.

Function:

```
FETCH(LOOKUP[COLUMN_NAME,FIELD_NAME,FIELD_VALUE,DESCR,LANGUAGE_CD])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
FIELD_NAME	Lookup Field Name	Lookup Field Value Language
FIELD_VALUE	Lookup Field Value	Lookup Field Value Language
DESCR	Description	Lookup Field Value Language
LANGUAGE_CD	Language Code	Lookup Field Value Language

List of output field/column values:

Output Parameter	Description	Table Description
FIELD_VALUE	Lookup Field Value	Lookup Field Value Language
DESCR	Description	Lookup Field Value Language
DESCRLONG	Detail Description	Lookup Field Value Language

11.1.11 Fetch Parent Person

These function can be used to get the root parent person from the person hierarchy

Account identifier can be retrieved for either the given *PER_ID_NBR* i.e. with reference to the defined relationship between Person and Account in "Account Person Relationship" table or *ACCT_NBR* i.e. with reference to the relationship defined in the "Maintains External Account Numbers" table.

Only input parameters *PER_ID_NBR* and *ID_TYPE_CD* are optional and remaining are mandatory for this function.

Function:

```
FETCH(PARENT_PER[COLUMN_NAME,PER_ID_NBR,ID_TYPE_CD,PER_REL_TYPE_CD,ACC
T_REL_TYPE_CD,ACCT_NBR,ACCT_NBR_TYPE_CD,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
PER_REL_TYPE_CD	Person Relationship Type Code	Person Identifier
ACCT_REL_TYPE_CD	Account Relationship Type	Account Person Relationship

ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PER_ID	Person ID (Returned only when PER_ID_NBR is passed as an function input)	Person Identifier
PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
PRIM_SW	Primary ID switch	Person Identifier
VERSION	Version	Person Identifier

11.1.12 Fetch Price Assignment

These function can be used to get the price assignment details for the given Owner ID or Price Item Code or Price Assign ID.

Function:

```
FETCH(PRICE_ASSIGNMENT[COLUMN_NAME,OWNER_ID,PRICEITEM_CD,PRICE_STATUS_
FLAG,CRR_DT,PRICE_ASGN_ID,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
OWNER_ID	Owner ID	Price Assignment
PRICEITEM_CD	Price Item	Price Assignment

PRICE_STATUS_FLAG	Pricing Status	Price Assignment
CRR_DT	This date will be compared to be greater than or equal to END DATE in Price Assignment	Price Assignment
PRICE_ASGN_ID	Price Assignment ID	Price Assignment
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PRICE_ASGN_ID	Price Assignment ID	Price Assignment
OWNER_ID	Owner ID	Price Assignment
PA_OWNER_TYPE_FLG	Price Assignment Owner Type Flag	Price Assignment
START_DT	Start Date	Price Assignment
END_DT	End Date	Price Assignment
PRICEITEM_CD	Price Item	Price Assignment
PRICE_CURRENCY_CD	Pricing Currency	Price Assignment
PRICE_STATUS_FLAG	Pricing Status	Price Assignment
RS_CD	Rate Schedule	Price Assignment
PA_TYPE_FLAG	Price Assignment Type	Price Assignment
TXN_RATING_CRITERIA	Transaction Rating Criteria	Price Assignment

11.1.13 Fetch Person

These function can be used to get the Person identifier with its corresponding detail field values i.e. below listed column/output values.

Account identifier can be retrieved for either the given *PER_ID_NBR* i.e. with reference to the defined relationship between Person and Account in "Account Person Relationship" table or *ACCT_NBR* i.e. with reference to the relationship defined in the "Maintains External Account Numbers" table.

Either of the inputs *PER_ID_NBR or ACCT_NBR* is mandatory input for this function.

Function:

```
FETCH(PER[COLUMN_NAME,PER_ID_NBR,ID_TYPE_CD,ACCT_REL_TYPE_CD,ACCT_NBR,
ACCT_NBR_TYPE_CD,DIVISION,ACCT_USAGE_FLG,PRIM_SW,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
ACCT_REL_TYPE_CD	Account Relationship Type	Account Person Relationship
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
PRIM_SW	Primary ID switch	Person Identifier
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PER_ID	Person ID	Person Identifier
ID_TYPE_CD	Person Identifier Type (Returned only when PER_ID_NBR is passed as an function input)	Person Identifier

ACCESS_GRP_CD	Access Group Code (Returned only when PER_ID_NBR is passed as an function input)	Person
PRIM_SW	Primary ID (Returned only when PER_ID_NBR is passed as an function input)	Person Identifier
ACCT_ID	Account ID (Returned only when ACCT_NBR is passed as an function input)	Account
CIS_DIVISION	Division of an Account (Returned only when ACCT_NBR is passed as an function input)	Account
CURRENCY_CD	Currency Code of an Account (Returned only when ACCT_NBR is passed as an function input)	Account
ACCT_REL_TYPE_CD	Account Relationship Type (Returned only when ACCT_NBR is passed as an function input)	Account Person Relationship
ACCT_NBR_TYPE_CD	Account Identifier Type (Returned only when ACCT_NBR is passed as an function input)	Maintains External Account Numbers
ENTITY_NAME	Contract Type (Returned only when ACCT_NBR is passed as an function input)	Person Name
ACCT_USAGE_FLG	Account Usage (Returned only when ACCT_NBR is passed as an function input)	Account
VERSION	Version	Person

11.1.14 Fetch Person Characteristic

These function can be used to get the Person identifier and Characteristics along with its other corresponding field values i.e. below listed column/output values.

Parameters *PER_ID_NBR and CHAR_TYPE_CD* are mandatory inputs for this function.

Function:

```
FETCH(PER_CHAR[COLUMN_NAME,PER_ID_NBR,ID_TYPE_CD,CHAR_TYPE_CD,PRIM_SW,
SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description

COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
CHAR_TYPE_CD	Characteristic Type Code	Account Person Relationship
PRIM_SW	Primary ID switch	Person Identifier
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PER_ID	Person ID	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
ACCESS_GRP_CD	Access Group Code	Person
PRIM_SW	Primary ID	Person Identifier
CHAR_TYPE_CD	Characteristic Type Code	Person Characteristics
CHAR_VAL	Characteristic Value	Person Characteristics
EFFDT	Effective Date	Person Characteristics
CHAR_VAL_FK1	Characteristic Value FK1	Person Characteristics
CHAR_VAL_FK2	Characteristic Value FK2	Person Characteristics
CHAR_VAL_FK3	Characteristic Value FK3	Person Characteristics
CHAR_VAL_FK4	Characteristic Value FK4	Person Characteristics
CHAR_VAL_FK5	Characteristic Value FK5	Person Characteristics
VERSION	Version	Person

11.1.15 Fetch Price Item Contract

These function can be used to get the price item for contract along with its other corresponding field values i.e. below listed column/output values.

Parameters *ACCT_NBR* is mandatory input for this function.

Function:

```
FETCH(PISA[COLUMN_NAME,ACCT_NBR,ACCT_NBR_TYPE_CD,PRICE_ITEM_CD,DIVISIO
N,ACCT_USAGE_FLG,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
PRIM_SW	Primary ID switch	Person Identifier
PRICE_ITEM_CD	Price Item	Pricing Item
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
ACCT_ID	Account ID	Account
CIS_DIVISION	Division of an Account	Account
CURRENCY_CD	Currency Code of an Account	Account

ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
ACCT_USAGE_FLG	Account Usage	Account
SA_ID	Contract ID	SA (Service Agreement)
SA_TYPE_CD	Contract Type	SA (Service Agreement)
VERSION	Version	SA (Service Agreement)

11.1.16 Fetch Price List

These function can be used to get the price list along with its other corresponding field values i.e. below listed column/output values.

Either of parameter *PRICELIST_ID or DESCR* is mandatory inputs for this function.

Function:

```
FETCH(PRICE_LIST[COLUMN_NAME,PRICELIST_ID,PL_TYPE,PL_STATUS_FLG,DESCR,
LANGUAGE_CD,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PRICELIST_ID	Price List ID	Price List
PL_TYPE	Price List Type	Price List
PL_STATUS_FLG	Status	Price List
DESCR	Description (This can have the wild card '%' to pass the partial description)	Price List
LANGUAGE_CD	Primary ID switch	Price List Language
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PRICELIST_ID	Price List ID	Price List
PL_TYPE	Price List Type	Price List
PL_STATUS_FLG	Status	Price List
DESCR	Description	Price List Language
PL_START_DT	Effective Start Date	Price List
PL_END_DT	Effective End Date	Price List
PL_GLOBAL_SW	Global Price List Switch	Price List

11.1.17 Fetch Price List Characteristic

These function can be used to get the Price List identifier and its Characteristics along with its other corresponding field values i.e. below listed column/output values.

Either of parameters **CHAR_VAL** *or* ***CHAR_TYPE_CD*** *or* ***ADHOC_CHAR_VAL*** is mandatory input for this function.

Function:

```
FETCH(PRICE_LIST_CHAR[COLUMN_NAME,CHAR_TYPE_CD,ADHOC_CHAR_VAL,CHAR_VAL
,PRICELIST_ID,PL_TYPE,PL_STATUS_FLG,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
CHAR_TYPE_CD	Characteristic Type Code	Price List Characteristic
ADHOC_CHAR_VAL	Adhoc Characteristic Value	Price List Characteristic
CHAR_VAL	Characteristic Value	Price List Characteristic
PRICELIST_ID	Price List ID	Price List
PL_TYPE	Price List Type	Price List
PL_STATUS_FLG	Status	Price List
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records.	

	If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PRICELIST_ID	Price List ID	Price List
PL_TYPE	Price List Type	Price List
PL_STATUS_FLG	Status	Price List
PL_START_DT	Effective Start Date	Price List
PL_END_DT	Effective End Date	Price List
PL_GLOBAL_SW	Global Price List Switch	Price List
CHAR_TYPE_CD	Characteristic Type Code	Price List Characteristics
CHAR_VAL	Characteristic Value	Price List Characteristics
EFFDT	Effective Date	Price List Characteristics
CHAR_VAL_FK1	Characteristic Value FK1	Price List Characteristics
CHAR_VAL_FK2	Characteristic Value FK2	Price List Characteristics
CHAR_VAL_FK3	Characteristic Value FK3	Price List Characteristics
CHAR_VAL_FK4	Characteristic Value FK4	Price List Characteristics
CHAR_VAL_FK5	Characteristic Value FK5	Price List Characteristics
VERSION	Version	Price List

11.1.18 Fetch Price List Assignment ID

These function can be used to get the active price list assignment identifier along with its other corresponding field values i.e. below listed column/output values.

Different combination of inputs can be used to invoke this function.

Either parameters **ACCT_NBR** and **ACCT_NBR_TYPE_CD** can be passed to get the **PARTY_UID** or directly pass the **PARTY_UID** input to get the required active price list assignment details.

Either parameters **DESCR** and **LANGUAGE_CD** can be passed to get the **PRICELIST_ID** or directly pass the **PRICELIST_ID** input to get the required active price list assignment details.

Function:

```
FETCH(PL_ASSGN[COLUMN_NAME,ACCT_NBR,ACCT_NBR_TYPE_CD,PARTY_UID,PRICELI
ST_ID,DESCR,LANGUAGE_CD,START_DT,END_DT, PL_STATUS_FLG,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
ACCT_NBR	Account Identifier	Maintains External Account Numbers
ACCT_NBR_TYPE_CD	Account Identifier Type	Maintains External Account Numbers
PARTY_UID	Party UID	Price List assignment
PRICELIST_ID	Price List ID	Price List
DESCR	Price List Description (This can have the wild card '%' to pass the partial description)	Price List Language
LANGUAGE_CD	Language Code (Default is set to 'ENG')	Price List Language
START_DT	Start Date (This date will be compared to be less than or equal to START DATE in Price List assignment)	Price List assignment
END_DT	End Date (This date will be compared to be greater than or equal to END DATE in Price List assignment)	Price List assignment
PL_STATUS_FLG	Status Flag. By default its value will be 'ACTV'.	Price List
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PRICELIST_ASGN_ID	Account ID	Price List assignment
PARTY_UID	Party UID	Price List assignment
PRICELIST_ID	Price List ID	Price List assignment

START_DT	Start Date	Price List assignment
END_DT	End Date	Price List assignment
BUS_OBJ_CD	Business Object Code	Price List assignment
SEQNO	Sequence No.	Price List assignment

11.1.19 Fetch Person Party UID

These function can be used to get and map the Person identifier, Party UID or its corresponding column values in ORMB system to the input payload i.e. manipulate the service payload.

Parameter **PER_ID_NBR** is mandatory input for this function.

Function:

```
FETCH(ACCT_PUID[COLUMN_NAME,PER_ID_NBR,ID_TYPE_CD,DIVISION,ACCT_USAGE_
FLG,PRIM_SW,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PER_ID_NBR	Person Id Number	Person Identifier
ID_TYPE_CD	Person Identifier Type	Person Identifier
DIVISION	Division	Account
ACCT_USAGE_FLG	Account Usage	Account
PRIM_SW	Primary ID switch	Person Identifier
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
CIS_DIVISION	Division of an Account	Account
CURRENCY_CD	Currency Code of an Account	Account
PER_ID	Person ID	Person Identifier
PARTY_UID	Party UID	Party
ACCT_REL_TYPE_CD	Account Relationship Type	Account Person Relationship
ACCT_USAGE_FLG	Account Usage	Account
VERSION	Version	Party

11.1.20 Fetch Person Relationship

These function can be used to get and map the Person to Person relationship with its corresponding column/output values listed below.

Parameter **PER_ID_NBR** is mandatory input for this function.

Function:

```
FETCH(PER_RELATION[COLUMN_NAME,PER_ID1,PER_ID2,PER_REL_TYPE_CD,FINAN_R
EL_SW,END_DT,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
PER_ID1	Person1	Person to Person
PER_ID2	Child Person	Person to Person
PER_REL_TYPE_CD	Relationship Type	Person to Person
FINAN_REL_SW	Financial Relationship	Person to Person
END_DT	End Date (This date will be compared to be greater than or equal to END DATE or retrieved if NULL)	Person to Person
SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level.	

	If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
PER_ID1	Person1	Person to Person
PER_ID2	Child Person	Person to Person
PER_REL_TYPE_CD	Relationship Type	Person to Person
FINAN_REL_SW	Financial Relationship	Person to Person
END_DT	End Date	Person to Person
VERSION	Version	Person to Person

11.1.21 Fetch Rate Map Id

These function can be used to get and map the Rate Component details with its other corresponding column/output values listed below.

Either combination of input parameters **RS_CD, RC_DESCR, RS_DESCR, RC_SEQ** can be passed or can directly pass input **RC_MAP_ID** to this function.

Function:

```
FETCH(RATE_COMPONENT[COLUMN_NAME,RS_CD,RC_DESCR,RS_DESCR,RC_SEQ,RC_MAP
_ID,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
RS_CD	Rate Schedule	Rate Component Map
RC_DESCR	Rate Component Description	Rate Component Language
RS_DESCR	Rate Schedule Description	Rate Schedule Language
RC_SEQ	Sequence	Rate Component Map
RC_MAP_ID	RC Map ID	Rate Component Map

SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
RS_CD	Rate Schedule	Rate Component Map
RC_DESCR	Rate Component Description	Rate Component Language
RS_DESCR	Rate Schedule Description	Rate Schedule Language
RC_SEQ	Sequence	Rate Component Map
RC_MAP_ID	RC Map ID	Rate Component Map

11.1.22 Fetch User

These function can be used to get and map the User details with its other corresponding column/output values listed below.

Input parameter **USER_ID** is mandatory to invoke this function.

Function:

```
FETCH(USER[COLUMN_NAME,USER_ID,EXT_USER_ID,USER_TYPE_FLG,USER_ENABLE_F
LG,OWNER_FLG,SET_OPERATION])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
COLUMN_NAME	Column name to be mapped from the returned RESULTSET. i.e. **OUTPUT** field name	
USER_ID	User ID	User
EXT_USER_ID	External User Id	User
USER_TYPE_FLG	User Type	User
USER_ENABLE_FLG	User Enable	User
OWNER_FLG	Owner	User

SET_OPERATION	This is a BOOLEAN value. If **TRUE** - Reset operation configured at service level. If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records. If **NULL** – Will do nothing but only throw an error if function return multiple records.	

List of output field/column values:

Output Parameter	Description	Table Description
USER_ID	User ID	User
FIRST_NAME	First Name	User
LAST_NAME	Last Name	User
TNDR_SOURCE_CD	Tender Source	User
DISP_PROF_CD	Display Profile Code	User
TD_ENTRY_AGE_DAYS1	Lower Age Limit for Yellow Bar	User
TD_ENTRY_AGE_DAYS2	Upper Age Limit for Yellow Bar	User
ACCESS_GRP_CD	Access Group	User
EMAILID	Email Address	User
USER_TYPE_FLG	User Type	User
DISP_ALL_PREM_SW	Display All Premises	User
OWNER_FLG	Owner	User
EXT_USER_ID	External User ID	User
TIME_ZONE_CD	Time Zone	User
USER_ENABLE_FLG	User Enable	User
VERSION	Version	User

11.1.23 Check Access Group

These function can be used to validate the input Access Group.

Input parameter **USER_ID** is mandatory to invoke this function.

Function:

```
FETCH(ACCESS_GRP[ACCESS_GRP])
```

Expected list of input field/column values:

Input Parameter	Description	Table Description
ACCESS_GRP	Access Group	Access Group

List of output field/column values:

Output Parameter	Description	Table Description
ACCESS_GRP	Access Group	Access Group

11.2 COPY function

Copy function can be used to get the required column value of any existing entity in ORMB system. This function will invoke the Business Object (BO) or Business Service (BS) that is provided as parameter value i.e. "**SERVICE_NAME**" and get the required column or list block i.e. "**OUTPUT_XPATH**" from its returned XML data payload.

COPY functions can be used in only **FIELD TRANSFORMATION** section but not in HEADER and FOOTER TRANSFORMATION.

COPY with **BO** – To invoke BO, parameter **SERVICE_TYPE** value must be **BO.** This function will require an input of primary key of a primary table defined in its corresponding Maintenance Object (MO) meta-.

COPY with **BS** – To invoke BS, parameter **SERVICE_TYPE** value must be **BS.** This function can have any of those possible inputs defined in BS HEADER schema.

Function:

```
COPY(SERVICE_TYPE, SERVICE_NAME, OUTPUT_XPATH, SET_OPERATION,
INPUT[ELEMENT_XPATH=INPUT_VALUE], PREPROCESS[OPERATION],
POSTPROCESS[OPERATION], FILTER[ELEMENT_XPATH=INPUT_VALUE],
UPD[ELEMENT_XPATH=INPUT_VALUE])
```

11.2.1 INPUT Parameter details:

SERVICE_TYPE

Type of Service to be invoked. It can have either of this two values "BO" or "BS".

SERVICE_NAME

ORMB "BO" or "BS" service names that is to be invoked for copying.

OUTPUT_XPATH

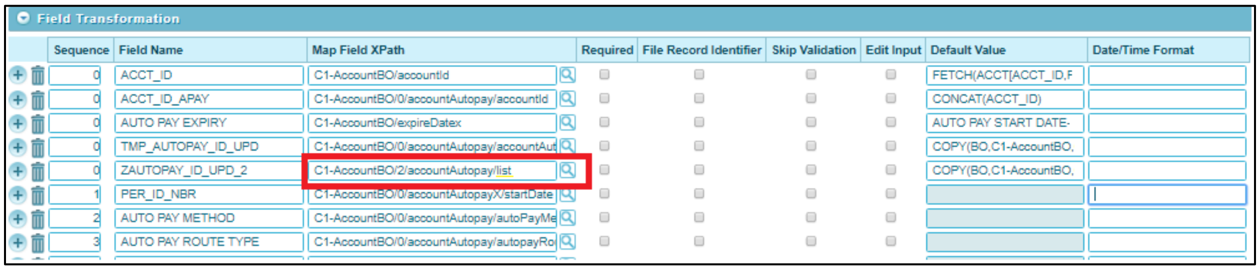
COPY function will be getting the response in XML form. So, this will be the XPATH of that element whose value we want to map from the response XML. This XPATH can be either of an individual element or the list block in the response XML.

List Block XPATH will allow to copy the complete list block from the response XML to the input payload.

Note – For list block copy we have to suffix the XPATH with "**/list**" in the corresponding 'MAP FIELD XPATH' column value.

Example:

COPY(BO,C1-AccountBO,**C1-AccountBO/accountAutopay**,FILTER[C1-AccountBO/accountAutopay/endDate=''],UPD[AUTO PAY EXPIRES ON= '2020-12-31',C1-AccountBO/accountAutopay/endDate=AUTO PAY EXPIRY])



Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Skip Validation	Edit Input	Default Value	Date/Time Format
0	ACCT_ID	C1-AccountBO/accountId	☐	☐	☐	☐	FETCH(ACCT[ACCT_ID,F	
0	ACCT_ID_APAY	C1-AccountBO/0/accountAutopay/accountId	☐	☐	☐	☐	CONCAT(ACCT_ID)	
0	AUTO PAY EXPIRY	C1-AccountBO/expireDatex	☐	☐	☐	☐	AUTO PAY START DATE-	
0	TMP_AUTOPAY_ID_UPD	C1-AccountBO/0/accountAutopay/accountAut	☐	☐	☐	☐	COPY(BO,C1-AccountBO,	
0	ZAUTOPAY_ID_UPD_2	C1-AccountBO/2/accountAutopay/list	☐	☐	☐	☐	COPY(BO,C1-AccountBO,	
1	PER_ID_NBR	C1-AccountBO/0/accountAutopayX/startDate	☐	☐	☐	☐		
2	AUTO PAY METHOD	C1-AccountBO/0/accountAutopay/autoPayMe	☐	☐	☐	☐		
3	AUTO PAY ROUTE TYPE	C1-AccountBO/0/accountAutopay/autopayRo	☐	☐	☐	☐		

Figure 75: Map Field XPath

In this above COPY function example, we are mapping the "C1-AccountBO/accountAutopay" list block from the copied entity to the input service payload.

COPY for list block retrieval will be invoked with having service payload XML as an input. This service payload must have the required primary key inputs for the BO to be invoked.

If we are invoking BO '**C1-AccountBO**', then, this input service payload must have '**C1-AccountBO/accountId'** element value i.e. Primary key value of primary table defined in Account MO.

SET_OPERATION

This can have a **BOOLEAN** value.

If **TRUE** - Reset operation configured at service level.

If **FALSE** - Validate the fetch function output i.e. Throws error if returns NULL or multiple records.

If **NULL** – Will do nothing but only throw an error if function return multiple records.

INPUT [ELEMENT_XPATH=INPUT_VALUE]

This is an optional input parameter block. Applicable only if we want to get the individual element value i.e. not the list block from response XML.

This block can have multiple comma-separated inputs.

- **ELEMENT_XPATH –** XPATH of the input element i.e. key XPATH in the BO/BS to be invoked
- **INPUT_VALUE –** Input element i.e. key value. This will be used to invoke the BO/BS

Input values can have LITERAL values i.e. static values enclosed in single quotes or the CONSTANT values i.e. ":BUS_DATE, :BUS_DTTM , :STD_DATE, :STD_DTTM, :SYSDATE, :SYS_DTTM, :CHAR_DATE, :FILENAME, :FILE_ID" or those configured FIELD names in DATA TRANSFORMATION corresponding to its section i.e. HEADER or FOOTER or FIELD TRANSFORMATION whose mapped value will be referred.

Example:

COPY(BO,CM-TaxRates,CM-TaxRates/lookupValue,true,**INPUT[CM-TaxRates/bo=BUS_OBJ_CD,CM-TaxRates/lookupValue=LOOKUPKEY]**)

PREPROCESS [OPERATION]

This is an optional parameter block. This can be used to perform the required pre-processing and then processing the actual record.

- **OPERATION –** This is optional parameter. It can have NULL value. If NULL, then, it will have the operation value that is configured in FRT at service level.
- **INPUT** block is mandatory for **PREPROCESS** and **POSTPROCESS.** Multiple records can be added or updated using FETCH function in the INPUT block. Same applies to **POSTPROCESS.**

e.g.
COPY(BO,C1_PERSON_BO,C1_PERSON_BO/personPerson,**INPUT[PARENT_PER_ID=FETCH(PER_RELATION[PER_ID1,null,CHILD_PER_ID,null,null,:BUS_DATE,false])]**,PREPROCESS[UPD],FILTER[CHILD_PER_ID=CHILD_PER_ID && C1_PERSON_BO/personPerson/endDate='' &&
C1_PERSON_BO/personPerson/personId1!=PARENT_PER_ID || CHILD_PER_ID=CHILD_PER_ID &&
C1_PERSON_BO/personPerson/endDate.ISAFTER(:BUS_DATE) &&
C1_PERSON_BO/personPerson/personId1!=PARENT_PER_ID],UPD[C1_PERSON_BO/personPerson/endDate=END_DATE_UPD])

POSTPROCESS [OPERATION]

It is an optional parameter block. This can be used to perform the required post-processing and then processing the actual record.

- **OPERATION –** This is optional parameter. It can have NULL value. If NULL, then, it will have the operation value that is configured in FRT at service level.

FILTER [ELEMENT_XPATH=INPUT_VALUE]

It is an optional parameter block. These can be used if we want to apply filter and get only the specific element value from the possible list of blocks in response XML.

- **ELEMENT_XPATH –** XPATH of the filter element i.e. filter key XPATH in the response XML.
- **INPUT_VALUE –** Filter condition value i.e. key value. This will be used to filter the value from response XML.

Input values can have LITERAL values i.e. static values enclosed in single quotes or the CONSTANT values i.e. ":BUS_DATE, :BUS_DTTM , :STD_DATE, :STD_DTTM, :SYSDATE, :SYS_DTTM, :CHAR_DATE, :FILENAME, :FILE_ID" or those configured FIELD names in DATA TRANSFORMATION corresponding to its section i.e. HEADER or FOOTER or FIELD TRANSFORMATION whose mapped value will be referred.

Filter criteria can be defined using an expression using supported conditional and logical operators.

e.g.
COPY(BO,C1_PERSON_BO,C1_PERSON_BO/personPerson,INPUT[PARENT_PER_ID=FETCH(PER_RELATION[PER_ID1,null,CHILD_PER_ID,null,null,:BUS_DATE,false])],PREPROCESS[UPD],**FILTER**[CHILD_PER_ID=CHI

LD_PER_ID && C1_PERSON_BO/personPerson/endDate='' &&
C1_PERSON_BO/personPerson/personId1!=PARENT_PER_ID || CHILD_PER_ID=CHILD_PER_ID &&
C1_PERSON_BO/personPerson/endDate.ISAFTER(:BUS_DATE) &&
C1_PERSON_BO/personPerson/personId1!=PARENT_PER_ID],UPD[C1_PERSON_BO/personPerson/endD
ate=END_DATE_UPD])

Equality, Relational, and Conditional Operators

The equality and relational operators determine if one operand is greater than, less than, equal to, or not equal to another operand.

List of Operators:

Operator	Description	Supported Data Type
=	Equal to	CHAR and NUMBER
!=	Not Equal to	CHAR and NUMBER
<	Less than	NUMBER
<=	Less than or Equal to	NUMBER
>	Greater than	NUMBER
>=	Greater than or Equal to	NUMBER
ISBEFORE	Less than	DATE
ISBEFOREOREQUAL	Less than or Equal to	DATE
ISAFTER	Greater than	DATE
ISAFTEROREQUAL	Greater than or Equal to	DATE
ISEQUAL	Equal to	DATE
ISNOTEQUAL	Not Equal to	DATE

Logical Operators:

The && and || operators perform Conditional-AND and Conditional-OR operations on two boolean expressions. These operators exhibit "short-circuiting" behavior, which means that the second operand is evaluated only if needed.

Note - No enclosing brackets allowed.

List of Logical Operators:

Operator	Description
&&	AND
||	OR

UPD[ELEMENT_XPATH=INPUT_VALUE]

This is an optional input parameter. These can be used to update the element value in the copied XML block.

Multiple element values can be updated in a copied XML block having the multiple comma-separated element key values.

- **ELEMENT_XPATH –** XPATH of the element in the copied XML response for which the value is to be updated.
- **INPUT_VALUE –** Input element i.e. key value. This will be used to update the element value in the copied XML block.

Input values can have LITERAL values i.e. static values enclosed in single quotes or the CONSTANT values i.e. ":BUS_DATE, :BUS_DTTM , :STD_DATE, :STD_DTTM, :SYSDATE, :SYS_DTTM, :CHAR_DATE, :FILENAME, :FILE_ID,:REMOVE,:REMOVE_FLD" or those configured FIELD names in DATA TRANSFORMATION corresponding to its section i.e. HEADER or FOOTER or FIELD TRANSFORMATION whose mapped value will be referred.

Eg. COPY(BO,C1-AccountBO,C1-AccountBO/accountAutopay,FILTER[C1-AccountBO/accountAutopay/endDate=''],**UPD**[AUTO PAY EXPIRES ON= '2020-12-31',C1-AccountBO/accountAutopay/endDate=AUTO PAY EXPIRY])

11.3 String Handling

String handling functions can be used for modifying the individual field inputs before formation of the input payload. These functions can also be used for validations in the SCRIPT function.

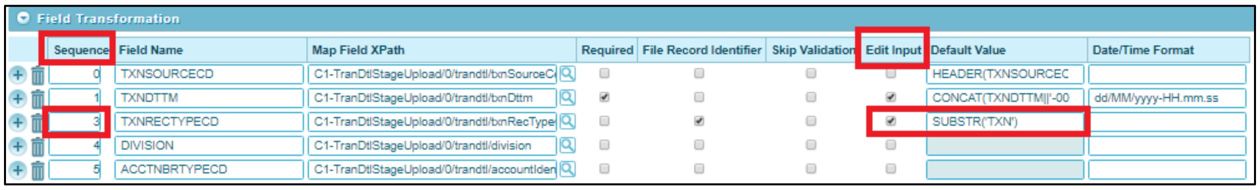
STRING functions can be used in all three sections of DATA TRANSFORMATION i.e. HEADER, FOOTER or FIELD.

String functions can be used in two ways:

- **Modify individual field input** –

These can be done only for the input file data i.e. not for DEFAULT values with 'SEQUENCE - 0'. Fields with SEQUENCE '0' can use this String functions only with SCRIPT function.

EDIT INPUT flag should be selected to have default values configured for FIELD's that have SEQUENCE not equal to ZERO i.e. '0'.



Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Skip Validation	Edit Input	Default Value	Date/Time Format
0	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC	☐	☐	☐	☐	HEADER(TXNSOURCEC	
1	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓	☐	☐	✓	CONCAT(TXNDTTM||'-00	dd/MM/yyyy-HH.mm.ss
3	TXNRECTYPECD	C1-TranDtlStageUpload/0/trandtl/txnRecType	☐	✓	☐	✓	SUBSTR('TXN')	
4	DIVISION	C1-TranDtlStageUpload/0/trandtl/division	☐	☐	☐	☐		
5	ACCTNBRTYPECD	C1-TranDtlStageUpload/0/trandtl/accountIden	☐	☐	☐	☐		

Figure 76: Field Transformation

- **Use with SCRIPT function** -

SCRIPT function with STRING functions use has no constraints. Invoking STRING function with in SCRIPT has a different SYNTAX. STRING functions are invoked with reference to the FIELD NAME i.e. **FIELD_NAME.STRING_FUNCTION**

Example:

SCRIPT([PERSONIDNUMBER!=''&&PERSONIDTYPE=''],{**PERSONIDNUMBER.SUBSTR(0,3)**})

List of functions:

- Substring
    - Substring from search string(Inclusive) - **SUBSTR('startString')**
    - Substring from startIndex - **SUBSTR(startIndex)**
    - Substring from search string(Inclusive) to endIndex - **SUBSTR('startString',endIndex)**
    - Substring from startIndex to search string(Inclusive) - **SUBSTR(startIndex,'endString')**
    - Substring from startIndex to endIndex - **SUBSTR(startIndex,endIndex)**
    - Substring between given string inputs - **SUBSTR('startString','endString')**
    - Substring to get last index Char Sequence - **SUBSTRLAST(startIndex)**
- Prefix the input string - **PREFIX('prefixString')**
- Suffix the input string - **SUFFIX('suffixString')**
- Convert string to uppercase - **UPPERCASE()**
- Convert string to lowercase - **LOWERCASE()**
- Right trim the input string - **RTRIM()**
- Left trim the input string - **LTRIM()**
- Right and Left trim the input string - **TRIM()**
- Concat - **CONCAT(FLD_NAME1||FLD_NAME2)**
- **ISNUMBER(INPUT_VALUE)** – This can be used only with SCRIPT function
- **LENGTH(INPUT_VALUE)** – This can be used only with SCRIPT function
- **ISUTF8 (INPUT_VALUE)** – This can be used only with SCRIPT function
- **CONTAINS (INPUT_VALUE,SEARCH_STRING,EXPECTED_OUTPUT)** – This can be used only with SCRIPT function
- **STARTSWITH(INPUT_FLD,SEARCH_STRING,EXPECTED_OUTPUT)–** This can be used only with SCRIPT function
- **ENDSWITH(INPUT_FLD,SEARCH_STRING,EXPECTED_OUTPUT)**– This can be used only with SCRIPT function

11.3.1 Substring

Returns a string that is a substring of the corresponding field input string.

Substring from search string

The substring begins from the characters that are specified as **startString** and extends to the end of this string.

Function: SUBSTR('startString')

Examples:

"unhappy" - SUBSTR('ha') returns "happy"

"Harbison" - SUBSTR('rbi') returns "rbison"

Parameters:

startString the starting string, inclusive. This input string should be enclosed in single quotes.

Returns:

The specified substring.

Substring from startIndex

The substring begins from the given startIndex and extends to the end of this string.

Function: SUBSTR(startIndex)

Examples:

"unhappy" - SUBSTR(2) returns "happy"

"Harbison" - SUBSTR(3) returns "bison"

Parameters:

startIndex the starting index, inclusive.

Returns:

The specified substring.

Substring from search string to endIndex

The substring begins from the character that are specified as **startString** and extends to the character at index **endIndex** – 1.

Function: SUBSTR('startString',endIndex)

Examples:

"unhappy" - SUBSTR('ha',5) returns "hap"

Parameters:

startString the starting string, inclusive. This input string should be enclosed in single quotes.

endIndex the end index, exclusive.

Returns:

The specified substring.

Substring from startIndex to search string

The substring begins from the given **startIndex** and extends till the given **endString** of this string.

Function: SUBSTR(startIndex,'endString')
Examples:

"unhappy" - SUBSTR(2, 'ppy') returns "happy"

"Harbison" - SUBSTR(3,‘so’) returns "rbiso"

Parameters:

startIndex the end index, inclusive.

endString the end string, inclusive. This input string should be enclosed in single quotes.

Returns:

The specified substring.

Substring from startIndex to endIndex

The substring begins at the specified **startIndex** and extends to the character at index **endIndex** - 1. Thus the length of the substring is **endIndex**- **startIndex**.

Function: SUBSTR(startIndex, endIndex)

Examples:

"hamburger" - SUBSTR(4, 8) returns "urge"
"smiles" - SUBSTR(1, 5) returns "mile"

Parameters:

startIndex the starting index, inclusive.
endIndex the ending index, exclusive.

Returns:

The specified substring.

Substring between given string inputs

The substring returns the string **between** given **startString** and **endString** i.e. String exclusive of the given string inputs.

Function: SUBSTR('startString', 'endString')

Examples:

"beautiful" - SUBSTR(‘ea’,’fu’) returns "uti"

Parameters:

startString the starting string, exclusive. This input string should be enclosed in single quotes.

endString the end string, exclusive. This input string should be enclosed in single quotes.

Returns:

The specified substring.

Substring to get last index Char Sequence

The substring returns this no. of last char sequence i.e. given **length_of_char_sequence** for the input string

Function: SUBSTRLAST (length_of_char_sequence)

Examples:

"beautiful" - SUBSTRLAST(5) returns "tiful"

Parameters:

length_of_char_sequence is the no. of char sequence from the end of String.

Returns:

The specified substring.

11.3.2 Prefix the input string

The string is prefixed with the characters in the specified **prefixString**.

Function: PREFIX('prefixString')

Examples:

"morning" - PREFIX ('this') returns " thismorning"

Parameters:

prefixString the prefix String. This input string should be enclosed in single quotes.

11.3.3 Substring from search string

The string is suffixed with the characters in the specified **suffixString**.

Function: SUFFIX('suffixString')

Examples:

"morning" - SUFFIX('tea') returns "morningtea"

Parameters:

suffixString the suffix string. This input string should be enclosed in single quotes.

11.3.4 Convert string to uppercase

Converts all of the characters in this String to upper case using the rules of the default locale.

Function: UPPERCASE ()

Examples:

"happy" - UPPERCASE() returns "HAPPY"

Returns:

The uppercase string.

11.3.5 Convert string to lowercase

Converts all of the characters in this String to lower case using the rules of the default locale.

Function: LOWERCASE ()

Examples:

"HappyBirthday" - LOWERCASE() returns "happybirthday"

Returns:

The lowercase string.

11.3.6 Right trim the input string

Returns a string whose value is this string, with any trailing whitespace removed.

This method may be used to trim whitespace from the end of a string.

Function: RTRIM()

Examples:

"   unhappy   " – RTRIM() returns "   unhappy"

Returns:

A string whose value is this string, with any trailing white space removed, or this string if it has no trailing white space.

11.3.7 Left trim the input string

Returns a string whose value is this string, with any leading whitespace removed.

This method may be used to trim whitespace from the start of a string.

Function: LTRIM()

Examples:

"   unhappy   " – LTRIM() returns "unhappy   "

Returns:

A string whose value is this string, with any leading white space removed, or this string if it has no leading white space.

11.3.8 Trim the input string

Returns a string whose value is this string, with any leading and trailing whitespace removed.

This method may be used to trim whitespace from the beginning and end of a string.

Function: TRIM()

Examples:

"   unhappy   " – TRIM() returns   "unhappy"

Returns:

A string whose value is this string, with any leading and trailing white space removed, or this string if it has no leading or trailing white space.

11.3.9 CONCAT

This can be used to CONCAT the number of TRANSFORMATION FIELD's or LITERAL values. LITERAL values will be enclosed in single quotes.

"||" double pipe is the delimiter used to define multiple input values in CONCAT function.

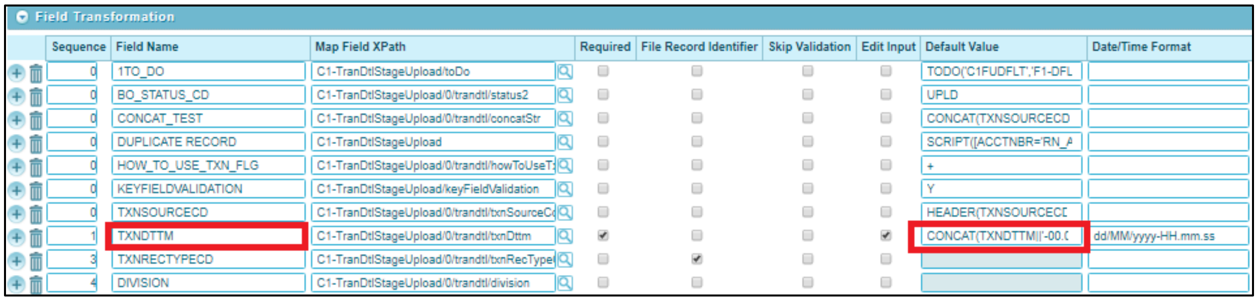
CONCAT function can be used in all three sections of DATA TRANSFORMATION i.e. HEADER, FOOTER or FIELD.

Function: CONCAT(FLD_NAME1||FLD_NAME2)

Examples:

If TXNDTTM = '2020-04-30'

CONCAT(TXNDTTM||'-00.00.00') returns '2020-04-30-00.00.00'



Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Skip Validation	Edit Input	Default Value	Date/Time Format
0	1TO_DO	C1-TranDtlStageUpload/toDo					TODO('C1FUDFLT','F1-DFL	
0	BO_STATUS_CD	C1-TranDtlStageUpload/0/trandtl/status2					UPLD	
0	CONCAT_TEST	C1-TranDtlStageUpload/0/trandtl/concatStr					CONCAT(TXNSOURCECD	
0	DUPLICATE RECORD	C1-TranDtlStageUpload					SCRIPT([ACCTNBR='RN_A	
0	HOW_TO_USE_TXN_FLG	C1-TranDtlStageUpload/0/trandtl/howToUseT					+	
0	KEYFIELDVALIDATION	C1-TranDtlStageUpload/keyFieldValidation					Y	
0	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC					HEADER(TXNSOURCECD	
1	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓			✓	CONCAT(TXNDTTM||'-00.(	dd/MM/yyyy-HH.mm.ss
3	TXNRECTYPECD	C1-TranDtlStageUpload/0/trandtl/txnRecType		✓				
4	DIVISION	C1-TranDtlStageUpload/0/trandtl/division						

Figure 77: Field Transformation TXNDTTM

Parameters:

This can be no. of FIELD names in TRANSFORMATION or the LITERAL values enclosed in single quotes.

Returns:

A string that represents the concatenation of the input parameters.

11.3.10 ISNUMBER

Returns true if and only if this string contains only numeric values. This function can be used only with **SCRIPT** function.

Function: ISNUMBER(INPUT_VALUE)

Examples:

- If **PERSONIDNUMBER** = '2020-04-30'
  SCRIPT([PERSONIDNUMBER!='' && **ISNUMBER(PERSONIDNUMBER)**],{PERSONIDNUMBER;''})

  Here, SCRIPT function returns **'' i.e. EMPTY_STRING**

- If **PERSONIDNUMBER** = '20200430'
  SCRIPT([PERSONIDNUMBER!='' && **ISNUMBER(PERSONIDNUMBER)**],{PERSONIDNUMBER;''})

  Here, SCRIPT function returns **PERSONIDNUMBER**

Parameters:

INPUT_VALUE the sequence to search for

Returns:

TRUE if this string contains only numeric values, **FALSE** otherwise

11.3.11 LENGTH

This can be used to compare the string length. It has to be used with conditional operators and numeric value. This function can be used only with **SCRIPT** function.

Function: LENGTH(INPUT_VALUE)

Examples:

- If **PERSONIDNUMBER** = '2020-04-30'
  SCRIPT([PERSONIDNUMBER!='' && **LENGTH(PERSONIDNUMBER)>15**],{PERSONIDNUMBER;''}

  Here, SCRIPT function returns **'' i.e. EMPTY_STRING**

- If **PERSONIDNUMBER** = '20200430'
  SCRIPT([PERSONIDNUMBER!='' && **LENGTH(PERSONIDNUMBER)=8**],{PERSONIDNUMBER;''})

  Here, SCRIPT function returns **PERSONIDNUMBER**

Parameters:

INPUT_VALUE the LENGTH to compare with

Returns:

TRUE if this LENGTH condition satisfies, **FALSE** otherwise

11.3.12 ISUTF8

Returns true if and only if this string contains only UTF8 character set. This function can be used only with **SCRIPT** function.

Function: ISUTF8(INPUT_VALUE)

Parameters:

INPUT_VALUE the character set to check for encoding

Returns:

TRUE if this string contains the string with UTF8 character set, **FALSE** otherwise.

11.3.13 CONTAINS

Returns true if and only if this string contains the specified sequence of char values. This function can be used only with **SCRIPT** function.

Function: CONTAINS(INPUT_FLD,SEARCH_STRING,EXPECTED_OUTPUT)

Examples:

- If **PERSONIDNUMBER** = '2020-04-30'

SCRIPT([PERSONIDNUMBER!=''&&CONTAINS(**PERSONIDNUMBER**,'2004',TRUE)],{PERSONIDNUMBER;''})

Here, SCRIPT function returns **'' i.e. EMPTY_STRING**

- If **PERSONIDNUMBER** = '20200430'

SCRIPT([PERSONIDNUMBER!=''&&CONTAINS(**PERSONIDNUMBER,**'2004',TRUE)],{PERSONIDNUMBER;''})

Here, SCRIPT function returns **PERSONIDNUMBER**

Parameters:

INPUT_FLD the input field for whose value we will have a search
SEARCH_STRING the sequence to search for
EXPECTED_OUTPUT the expected result. This can be TRUE or FALSE.

Returns:
TRUE if this returns the **EXPECTED_OUTPUT**, **FALSE** otherwise.

11.3.14 STARTSWITH

Returns true if and only if this string starts with the specified sequence of char values. This function can be used only with **SCRIPT** function.

Function: STARTSWITH(INPUT_FLD,SEARCH_STRING,EXPECTED_OUTPUT)

Examples:

- If **PERSONIDNUMBER** = '2020-04-30'

SCRIPT([PERSONIDNUMBER!=''&&STARTSWITH(**PERSONIDNUMBER**,'202004',TRUE)],{PERSONIDNUMBER;''})

Here, SCRIPT function returns **'' i.e. EMPTY_STRING**

- If **PERSONIDNUMBER** = '20200430'

SCRIPT([PERSONIDNUMBER!=''&&STARTSWITH(**PERSONIDNUMBER**'202004',TRUE)],{PERSONIDNUMBER;''})

Here, SCRIPT function returns **PERSONIDNUMBER**

Parameters:

INPUT_FLD the input field value to be checked for
SEARCH_STRING the sequence to search for
EXPECTED_OUTPUT the expected result. This can be TRUE or FALSE.

Returns:
TRUE if this returns the **EXPECTED_OUTPUT**, **FALSE** otherwise.

11.3.15 ENDSWITH

Returns true if and only if this string ends with the specified sequence of char values. This function can be used only with **SCRIPT** function.

Function: ENDSWITH(INPUT_FLD,SEARCH_STRING,EXPECTED_OUTPUT)

Examples:

- If **PERSONIDNUMBER** = '2020-04-30'

SCRIPT([PERSONIDNUMBER!=''&&ENDSWITH(**PERSONIDNUMBER**,'0430',TRUE)],{PERSONIDNUMBER;''})

Here, SCRIPT function returns **'' i.e. EMPTY_STRING**

- If **PERSONIDNUMBER** = '20200430'

SCRIPT([PERSONIDNUMBER!=''&&STARTSWITH(**PERSONIDNUMBER**'0430',TRUE)],{PERSONIDNUMBER;''})

Here, SCRIPT function returns **PERSONIDNUMBER**

Parameters:

INPUT_FLD the input field value to be checked for
SEARCH_STRING the sequence to search for
EXPECTED_OUTPUT the expected result. This can be TRUE or FALSE.

Returns:
TRUE if this returns the **EXPECTED_OUTPUT**, **FALSE** otherwise

11.4 SCRIPT function

The statements inside your **SCRIPT** function **SQUARE BRACKET i.e. [….]** are generally executed from left to right, in the order that they appear. **Control flow statements**, however, break up the flow of execution by employing decision making, enabling your program to conditionally execute particular blocks of code inside **SCRIPT** function **CURLY BRACES i.e. {….}**. This section describes the decision-making statements (**if-then, if-then-else**).

Multiple statements are separated with **SEMI-COLON** i.e**. ";" delimiter**.

SCRIPT function is categorized into two blocks:

- Control Statement Block – enclosed in ***SQUARE BRACKET i.e. [….]***
- Execution Block – enclosed in ***CURLY BRACES i.e. {….}***

Example:

SCRIPT([CHARACTERISTICTYPE2='CMSNDCRB'; CHARACTERISTICVALUE2!='' && CHARACTERISTICTYPE2=CHARACTERISTICVALUE2],{EFFECTIVEDATE1;EFFECTIVEDATE2; **:BUS_DATE**})

Execution work flow will be like,

```
if (CHARACTERISTICTYPE2='CMSNDCRB'){
        EFFECTIVEDATE1
}
else if (CHARACTERISTICVALUE2!='' && CHARACTERISTICTYPE2=CHARACTERISTICVALUE2){
        EFFECTIVEDATE2
}
else{
        :BUS_DATE
}
```

Equality, Relational, and Conditional Operators

The equality and relational operators determine if one operand is greater than, less than, equal to, or not equal to another operand.

List of Operators allowed in SCRIPT function control statements:

Operator	Description	Supported Data Type
=	Equal to	CHAR and NUMBER
!=	Not Equal to	CHAR and NUMBER
<	Less than	NUMBER
<=	Less than or Equal to	NUMBER
>	Greater than	NUMBER
>=	Greater than or Equal to	NUMBER
ISBEFORE	Less than	DATE

ISBEFOREOREQUAL	Less than or Equal to	DATE
ISAFTER	Greater than	DATE
ISAFTEROREQUAL	Greater than or Equal to	DATE
ISEQUAL	Equal to	DATE
ISNOTEQUAL	Not Equal to	DATE

Logical Operators:

The && and || operators perform Conditional-AND and Conditional-OR operations on two boolean expressions. These operators exhibit "short-circuiting" behavior, which means that the second operand is evaluated only if needed.

Note - No enclosing brackets allowed.

List of Logical Operators allowed in SCRIPT function control statement expressions:

Operator	Description
&&	AND
||	OR

Different types of inputs that can be used as input parameters in SCRIPT function are,

- **FETCH** function – This can be used in both blocks i.e. Control Statement Block and Execution Block

Example: FETCH function in Control Statement Block i.e. in [….]:

SCRIPT([FETCH(LOOKUP[DESCR,'C2_AGCY_CHNG_CARD_LKP',null,ACCT_NBR_FILE%,'ENG'])!=''],{:H1;ACCT_NBR_FILE})

SCRIPT workflow: IF the FETCH function is not empty THEN **:H1** (this is the value returned by FETCH function) ELSE **ACCT_NBR_FILE**

Example: FETCH function in Execution Block i.e. in {…}:

SCRIPT([ACCT_ID_1!=''],{FETCH(ACCT[ACCT_ID,NULL,NULL,ACCT ID TYPE,CUSTODY_ACCT_ID_VAL,ACCT ID TYPE,NULL,NULL,FALSE]); '123456789'})

SCRIPT workflow: IF the **ACCT_ID_1** is not empty THEN **execute** FETCH function and map the returned value. ELSE **123456789**

- **COPY** function – This can be used only in Execution Block i.e. {…}

Example:

SCRIPT([ACCT_ID_1!=''],{COPY(BO,C1-AccountBO,C1-AccountBO/accountAutopay,FILTER[C1-AccountBO/accountAutopay/endDate=''],UPD[AUTO PAY EXPIRES ON=AUTO PAY EXPIRY,C1-AccountBO/accountAutopay/endDate=AUTO PAY EXPIRY]); '123456789'})

SCRIPT workflow: IF the **ACCT_ID_1** is not empty THEN execute **COPY** function and map the returned value. ELSE **123456789**

- **DUPLICATE** function – This can be used only in Execution Block i.e. {…}

Example:

SCRIPT([ACCT_ID_1!=''],{DUPLICATE(ZACCT_ID_CUST=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_0_UPD=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_1_UPD=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_2_UPD=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_3_UPD=ZCASHACCT_ID_UPD); '123456789'})

SCRIPT workflow: IF the **ACCT_ID_1** is not empty THEN execute **DUPLICATE** function and map the returned value. ELSE **123456789**

- **CONCAT** function – This can be used only in Execution Block i.e. {…}

Example:
SCRIPT([TXNSOURCECD!=''],{CONCAT(TXNSOURCECD.SUBSTR(0,2)||TXNRECTYPECD.SUBSTRLAST(2)); '123456789'})

SCRIPT workflow: **IF** the **TXNSOURCECD** is not empty THEN execute **CONCAT** function and map the returned value. **ELSE** 123456789

- **ERROR(MSG_CAT,MSG_NUM,PARAM_LIST) –** It will be used if we want to throw an validation error for any control statement. This can be used only in Execution Block i.e. {…}

Example:

SCRIPT([EFFECTIVEDATE.isAfter(:CHAR_DATE)],{ERROR(90003,282,PERSONIDNUMBER,:CHAR_DATE,'test ')})

SCRIPT workflow: **IF** the **EFFECTIVEDATE** is after **:CHAR_DATE** THEN throw validation **ERROR.**

- **INPUT** parameters can be the **Literal** values i.e. static values enclosed in single quotes or the **Constant** values i.e. ":BUS_DATE, :BUS_DTTM , :STD_DATE, :STD_DTTM, :SYSDATE, :SYS_DTTM, :CHAR_DATE, :FILENAME, :FILE_ID,:REMOVE,:REMOVE_FLD,:SKIP" or those **configured Field names** in DATA TRANSFORMATION whose mapped value will be referred.
  - **:REMOVE** – For a conditional statement it can be used to remove the list block from the input payload.
  - **:REMOVE_FLD** – For a conditional statement it can be used to remove the attribute or element from the input payload.
  - **:SKIP** – For a conditional statement it can be used to skip that service execution.

11.5 TO DO Generation

This function can be used to generate TO DO for failure or success of each individual record execution status or a single TODO after completion of batch to get the summary of all the failures or multiple TODO's for each of the selected failures in this particular execution.

TODO functions can be used in all three sections of DATA TRANSFORMATION i.e. HEADER, FOOTER or FIELD.

Function:
TODO(TODO_TYPE,TODO_ROLE,ASSIGN_USER,STATUS,SINGLE_TODO,SORT_KEY[KEY_LIST],DRILL_KEY[KEY_LIST],CHAR[TYPE=VALUE],MSG[MSG_CAT,MSG_NUM,parm1,param2..])

TO DO generation is allowed at two levels,

- **RECORD level**: Parameter value of '**STATUS'** should not be **NULL**. It must be either **TRUE** or **FALSE.**
  - **TODO** for **FAILURE**: Parameter value of '**STATUS'** should be **FALSE**
  - **TODO** for **SUCCESS**: Parameter value of '**STATUS'** should be **TRUE**
- **BATCH level**: Parameter value of '**STATUS'** should be **NULL**.

INPUT Parameters:

- **TODO_TYPE** – TODO Type for which TODO is to be created
- **TODO_ROLE** – TODO Role will be responsible to work on this TODO
- **ASSIGN_USER** – Created TODO will be assigned to this USER
- **STATUS** – Possible values are TRUE, FALSE or NULL.
  - If **TRUE –** TODO will be created after **successful** execution of each record.
  - If **FALSE –** TODO will be created after **failed** execution of each record.
  - If **NULL –** TODO will be created after success or failed execution of this batch.
- **SINGLE_TODO –** This parameter will be used to decide if this TODO is applicable for all the defined SERVICES in this FRT. Possible values are **TRUE** or **FALSE**. Parameter in use if FRT has multiple SERVICES configured.
  - If **TRUE** – This TODO function will be applicable for TODO generation of all the SERVICES in this FRT
  - If **FALSE** – This TODO function will be applicable for TODO generation in only that SERVICE mentioned in **MAP FILED XPATH** value against which this function is configured.
- **SORT_KEY[KEY_LIST]** – Sort Key required for the TODO TYPE. Multiple keys i.e. COMMA-SEPARATED can be passed. This is optional parameter block.

Example:

TODO('C1FUDFLT','F1-DFLT',null,NULL,TRUE,SORT_KEY[:FILE_ID],DRILL_KEY[:FILE_ID,:FILENAME],CHAR['ZZACCT-E'='9901765140'],MSG[90003,302,:FILE_ID,:FILENAME,:BATCH_CD,:BATCH_NUM])

- **DRILL_KEY[KEY_LIST]** – Drill Key required for the TODO TYPE. Multiple keys i.e. COMMA-SEPARATED can be passed. This is optional parameter block.

Example:

TODO('C1FUDFLT','F1-DFLT',null,NULL,TRUE,SORT_KEY[:FILE_ID],DRILL_KEY[:FILE_ID,:FILENAME],CHAR['ZZACCT-E'='9901765140'],MSG[90003,302,:FILE_ID,:FILENAME,:BATCH_CD,:BATCH_NUM])

- **MSG[MSG_CAT,MSG_NUM,PARAM_LIST]–** Message for which the TODO generation should be done. Multiple keys i.e. COMMA-SEPARATED can be passed. This is optional parameter block.

INPUT parameters – These are the MESSAGE meta-data defined in ORMB.

MSG_CAT – Message Category
MSG_NUM – Message Number
PARAM_LIST – Input parameter list (COMMA-SEPARATED) for this message
Example:

TODO('C1FUDFLT','F1-DFLT',null,NULL,TRUE,SORT_KEY[:FILE_ID],DRILL_KEY[:FILE_ID,:FILENAME],CHAR['ZZACCT-E'='9901765140'],MSG[90003,302,:FILE_ID,:FILENAME,:BATCH_CD,:BATCH_NUM])

TODO generation has two behaviours for MSG block in record failures,

- TODO function **without** MSG Block
  - **Record Level** - TODO will be created after every **failed** execution of each record.
  - **Batch Level - S**ingle TODO will be created after completion of batch with the summary of all the failure details.
- TODO function **with** MSG Block
  - **Record Level** - TODO will be created after failure for the mentioned Category and Number execution of each record.
  - **Batch Level - S**ingle TODO will be created after completion of batch only for the mentioned Category and Number.

Note: Input parameters can have Literal values i.e. **Static** values enclosed in single quotes or **Constant** values, i.e. ":BUS_DATE, :BUS_DTTM , :STD_DATE, :STD_DTTM, :SYSDATE, :SYS_DTTM, :CHAR_DATE, :FILENAME, :FILE_ID" or those configured Field names in Data Transformation whose mapped value will be referred.

11.6 EVAL function

This function can be used to evaluate the mathematical expression. It supports expressions that are supported in JavaScript. For evaluating expression we are using JavaScript engine.

EVAL functions can be used in all three sections of DATA TRANSFORMATION i.e. HEADER, FOOTER or FIELD.

Function: EVAL([EXPRESSION];INPUT[PARAM_LIST_USED_IN_EXPRESSION])

INPUT parameters has two sections separated by semicolon (**;**) delimiter.

- **EXPRESSION** – This will be the expression that we want to evaluate. It is enclosed with square brackets i.e. (**[]**)

EVAL([(FLD1+FLD2+10)/2];INPUT[FLD1,FLD2])

- **PARAM_LIST_USED_IN_EXPRESSION** – This will be the list of Field Names (only those defined in Field Transformation) used in expression. It is enclosed with square brackets i.e. **[]**

EVAL([(FLD1+FLD2+10)/2];INPUT[FLD1,FLD2])

If FLD1 = 5 and FLD2=7 then this function will return 11

11.7 DUPLICATE function

This function can be used to create the duplicate record of only those required processed record in a file with also having the updates in that record to be duplicated. This function has to be always used with SCRIPT function. In SCRIPT function this can be used only in Execution Block i.e. {…}

DUPLICATE functions can be used only in Field Transformation i.e. not in HEADER or FOOTER.

Example:

SCRIPT([ACCT_ID_1!=''],{DUPLICATE(ZACCT_ID_CUST=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_0_UPD=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_1_UPD=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_2_UPD=ZCASHACCT_ID_UPD,ZACCTCHAR_CUST_3_UPD=ZCASHACCT_ID_UPD); '123456789'})

SCRIPT workflow: IF the **ACCT_ID_1** is not empty THEN execute **DUPLICATE** function and map the returned value. ELSE **123456789**

Function:
TODO(TODO_TYPE,TODO_ROLE,ASSIGN_USER,STATUS,SINGLE_TODO,SORT_KEY[KEY_LIST],DRILL_KEY[KEY_LIST],CHAR[TYPE=VALUE],MSG[MSG_CAT,MSG_NUM,parm1,param2..])

INPUT Parameter – This can have list of COMMA-SEPARATED KEY-VALUE pairs that is to be updated in duplicated record payload.

- **KEY** – This is the XPATH of element in the record payload for which the value is to be updated. It can be the FIELD NAME configured in FIELD TRANSFORMATION whose corresponding XPATH will be referred or we can directly have the XPATH as a KEY.
- **VALUE** – It can have the LITERAL values i.e. static values enclosed in single quotes or the CONSTANT values i.e. ":BUS_DATE, :BUS_DTTM , :STD_DATE, :STD_DTTM, :SYSDATE, :SYS_DTTM, :CHAR_DATE, :FILENAME, :FILE_ID" or those configured FIELD names in FIELD TRANSFORMATION whose mapped value will be referred.

11.8 Get Feature Configuration value

This function can be used to get the feature configuration value.

CONFIG function can be used in all three sections of DATA TRANSFORMATION i.e. HEADER, FOOTER or FIELD.

Function: CONFIG(featureConfigName, optionType, sequence)

- **featureConfigName** – Feature configuration name defined in ORMB system
- **optionType** – Option Type in feature configuration
- **sequence** – Sequence Number in feature configuration

INPUT parameters can have **Literal** values i.e. static values enclosed in single quotes or those configured Field names in Data Transformation whose mapped value will be referred.

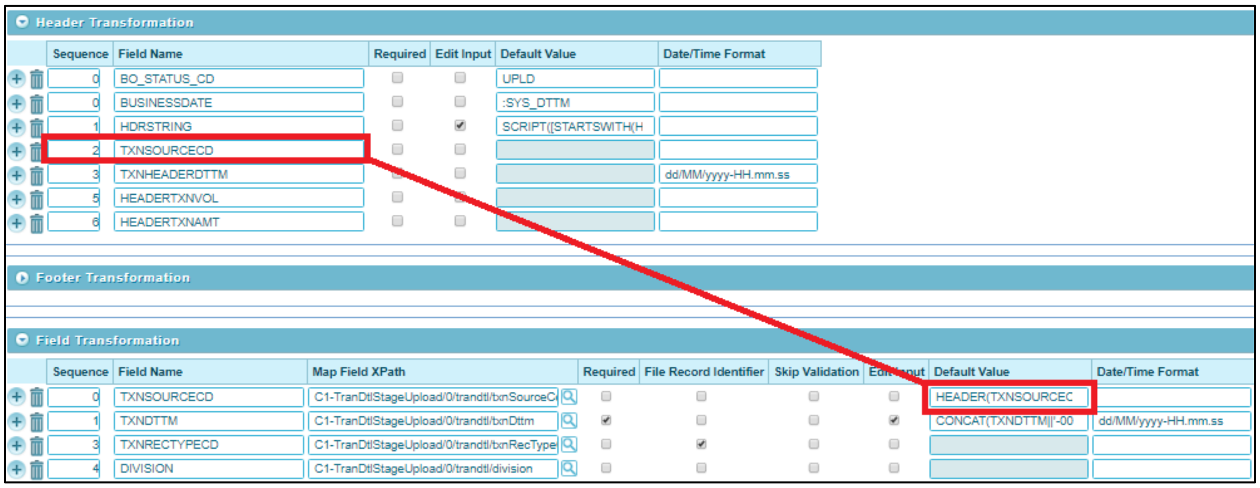
11.9 HEADER function

This function can be used to get the HEADER value referred in FIELD TRANSFORMATION section. HEADER function can be used only in FIELD TRANSFORMATION i.e. not in HEADER, FOOTER.

Function: HEADER(INPUT_HEADER_FIELD_NAME)

INPUT_HEADER_FIELD_NAME – Only FIELD names configured in HEADER TRANSFORMATION can be referred.

Example: HEADER(TXNSOURCECD)


Header Transformation

Sequence	Field Name	Required	Edit Input	Default Value	Date/Time Format
0	BO_STATUS_CD	☐	☐	UPLD	
0	BUSINESSDATE	☐	☐	:SYS_DTTM	
1	HDRSTRING	☐	✓	SCRIPT([STARTSWITH(H	
2	TXNSOURCECD	☐	☐		
3	TXNHEADERDTTM	☐	☐		dd/MM/yyyy-HH.mm.ss
5	HEADERTXNVOL	☐	☐		
6	HEADERTXNAMT	☐	☐		

Footer Transformation

Field Transformation

Sequence	Field Name	Map Field XPath	Required	File Record Identifier	Skip Validation	Edit Input	Default Value	Date/Time Format
0	TXNSOURCECD	C1-TranDtlStageUpload/0/trandtl/txnSourceC	☐	☐	☐	☐	HEADER(TXNSOURCEC	
1	TXNDTTM	C1-TranDtlStageUpload/0/trandtl/txnDttm	✓	☐	☐	✓	CONCAT(TXNDTTM||'-00	dd/MM/yyyy-HH.mm.ss
3	TXNRECTYPECD	C1-TranDtlStageUpload/0/trandtl/txnRecType	☐	✓	☐	☐		
4	DIVISION	C1-TranDtlStageUpload/0/trandtl/division	☐	☐	☐	☐		

Figure 78: HEADER function